国外工业控制与智能制造丛书





仿真建模与分析

（原书第5版）

[美] 埃弗里尔 M. 劳（Averill M. Law）著

范文慧 肖田元 等译

Simulation Modeling and Analysis
Fifth Edition

SIMULATION MODELING AND ANALYSIS

Averill M. Law



国外工业控制与智能制造丛书

仿真建模与分析

（原书第5版）

[美] 埃弗里尔 M. 劳（Averill M. Law）著
范文慧 肖田元 等译

Simulation Modeling and Analysis
Fifth Edition

SIMULATION MODELING ANALYSIS

Averill M. Law

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

仿真建模与分析（原书第 5 版）/（美）埃弗里尔 M. 劳（Averill M. Law）著；范文慧等译．—北京：机械工业出版社，2016.12
（国外工业控制与智能制造丛书）
书名原文：Simulation Modeling and Analysis, 5E

ISBN 978-7-111-55746-3

I. 仿… II. ①埃… ②范… III. ①离散事件系统 – 系统仿真 ②离散事件系统 – 系统建模 IV. TP271

中国版本图书馆 CIP 数据核字（2016）第 311243 号

本书版权登记号：图字：01-2014-7263

本书对离散事件系统仿真的重要方面进行了阐述，主要内容包括仿真建模入门、复杂系统建模、仿真软件、基础概率知识、仿真模型的建立、输入概率分布的选择、随机数发生器、随机变数的产生、单系统输出数据的分析、不同系统配置的比较、方差的缩减、实验设计与优化，以及基于 Agent 的仿真和系统动力学等。本书适合作为工程、计算机科学等相关专业的仿真课程教材，也可以供相关专业人士使用。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：王 颖 谢晓芳　　责任校对：董纪丽
印 刷：中国电影出版社印刷厂　　版 次：2017 年 1 月第 1 版第 1 次印刷
开 本：185mm×260mm 1/16　　印 张：32
书 号：ISBN 978-7-111-55746-3　　定 价：119.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991 88361066　　投稿热线：(010) 88379604
购书热线：(010) 68326294 88379649 68995259　　读者信箱：hzjsj@hzbook.com

出版者的话

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，信息学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的信息产业发展迅猛，对专业人才的需求日益迫切。这对我国教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其信息科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀教材将对我国教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson、McGraw-Hill、Elsevier、John Wiley & Sons、CRC、Springer 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Alan V. Oppenheim Thomas L. Floyd、Charles K. Alexander、Behzad Razavi、John G. Proakis、Stephen Brown、Allan R. Hambley、Albert Malvino、Peter Wilson 、H. Vincent Poor、Hassan K. Khalil、Gene F. Franklin、Rex Miller 等大师名家的经典教材，以“国外电子与电气技术丛书”和“国外工业控制与智能制造丛书”为系列出版，供读者学习、研究及珍藏。这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也越来越多被实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着电气与电子信息学科建设的不断完善和教材改革的逐渐深化，教育界对国外电气与电子信息教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www. hzbook. com
电子邮件：hzjsj@hzbook. com
联系电话：(010)88379604
联系地址：北京市西城区百万庄南街 1 号
邮政编码：100037



华章科技图书出版中心

译 者 序

埃弗里尔 M. 劳的《Simulation Modeling and Analysis》从 1982 年的第 1 版问世以来，不断修订发行，1991 年的第 2 版、2000 年的第 3 版、2007 年的第 4 版相继出版，2015 年修订与发行了第 5 版。本书的每一版都在前一版的基础上结合建模与仿真技术的发展不断地进行扩充与完善，是许多美国大学相关课程教学的主要参考书之一，也是进入该领域的研究人员的自学教材之一。

如前言所述，本书第 5 版的目标和前 4 版仍然一致：提供仿真研究的所有重要方面综合的与最新的进展，包括建模、仿真软件、模型校验和确认、输入建模、随机数发生器、随机变量和过程的产生、统计设计和仿真实验的分析，以及强调像制造这样最重要的应用领域。本书力图促进读者直观理解仿真与建模，并以技术上正确但更清晰的方式来阐述它们。全书有许多例子和习题，并有广泛的仿真参考文献和著作供读者进一步研究。

本书着眼于离散事件系统建模与仿真的原理、方法学的阐述，从概念出发，由浅入深，结合实际例子加以阐述与讨论，而不依赖于某一种商业化软件，因此适于作为专业课程的教材使用。对用本书作为教材的教师来说，可以从网址 www.mhhe.com/law 下载各种技术支持资料[⊖]，包括习题答案的全集、仿真模型和随机数发生器的相关代码。

机械工业出版社华章分社王颖编审根据本书在国内外的影响，决定引进第 5 版，并委托我们完成本书的翻译工作。参与本书翻译的人员是清华大学自动化系的部分教师与研究生，他们是范文慧、肖田元、王丽萍、徐炜达、部震霄、马成、李犁、孙宏波、刘雁兵等，全书由范文慧校核，最后由范文慧审定。限于译者的水平，书中的文字表达可能有不尽完善和不当之处，敬请读者批评指正。

译 者

⊖ 关于本书教辅资源，用书教师可向麦格劳·希尔教育出版公司北京代表处申请，电话：8008101936/010-62790299-108，电子邮件：instructorchina@mcgraw-hill.com。——编辑注

作者简介

埃弗里尔 M. 劳是 Averill M. Law & Associates 公司(亚利桑那州图森市)总裁，该公司专门从事仿真培训、咨询并提供软件。他以前是亚利桑那大学决策科学专业教授和威斯康星大学工业工程专业副教授。他在加州大学伯克利分校获得工业工程与运筹学硕士和博士学位，在加州大学长滩分校获得数学硕士学位，在宾夕法尼亚州立大学获得数学学士学位。

劳博士在 19 个国家讲过超过 525 个“仿真与统计”短期课程，包括许多内部讲座，如美铝(ALCOA)公司、AT&T 公司、波音公司、卡特彼勒公司、可口可乐公司、CSX 公司、加拿大国防研究与开发组织、通用电气公司、通用汽车公司、IBM 公司、Intel 公司、洛克西德·马丁公司、洛斯·阿尔莫斯国家实验室、美国导弹防御局、摩托罗拉公司、NASA、美国国家安全局、北大西洋公约组织(NATO)、诺斯罗普·格鲁门(Northrop Grumman)公司、挪威国防研究基础设施部、南非萨索尔公司、3M 公司、时代华纳公司、UPS 公司、美国空军部、美国陆军部、驻韩美军部、美国海军部、威瑞森公司、惠而浦(Whirlpool)公司，以及施乐公司。劳博士一直是许多机构的仿真顾问，例如埃森哲(Accenture)公司、波音公司、博思艾伦(BoozAllen & Hamilton)公司、康菲石油(ConocoPhillips)公司、美国国防建模与仿真办公室、惠普(Hewlett-Packard)公司、凯撒铝业(Kaiser Aluminum)公司、金伯利·克拉克(Kimberly-Clark)公司、M&M/Mars 公司、美国科学应用国际公司(SAIC)、桑迪亚国家实验室(Sandia National Labs)、瑞典国防材料管理部(Swedish Defense Materiel Administration)、3M 公司、Tropicana 公司、美国空军部、美国陆军部、美国潜艇公司、美国海军部、美国退伍军人管理局(Veteran's Administration)，以及施乐公司。

他是 ExpertFit 分布拟合软件的开发者，该软件自动进行仿真输入概率分布选择。ExpertFit在全世界多于 2000 个单位得到使用。他还开发了《Simulation of Manufacturing Systems》(制造系统仿真)与《How to Conduct a Successful Simulation Study》(怎样进行成功的仿真研究)的录像带。

2009 年，劳博士获得 INFORMS 仿真学会终身成就奖，撰写了三本书，发表了大量文章，涉及领域包括仿真、运筹学、统计学、制造和通信网络。他的文章“仿真输出数据的统计分析”(Statistical Analysis of Simulation Output Data)曾经是仿真领域《Operations Research》期刊上的第一篇特邀文章。他关于制造系统仿真的系列论文获 1988 年工业工程最佳论文奖。在其学术生涯期间，美国海军研究办公室支持其仿真研究连续达 8 年之久。他曾任 INFORMS 仿真学院院长。在 1990 年和 1991 年间，他为《Industrial Engineering》杂志撰写过关于仿真的专栏，也在国际仿真会议上做过大会主题发言。

符　号　表

符号或缩写	符号或缩写	符号或缩写	符号或缩写
A_i	FITA	$M/G/1$	SFD
ABS	$f(x)$	$M/M/1$	Skart
AR，ARMA	$F(x)$	$M/M/2$	(s, S)
ARTA	$f(x, y)$	$M/M/s$	S_i
ASAP3	F^{-1}	MLE	$S^2(n)$
AV	$\Gamma(\alpha,\beta)$	MRG	SME
A^T	geom(p)	MRG32k3a	t_i
Δb	GFSR	MSCO	$t_{n-1,1-\alpha/2}$
Bernoulli(p)	$GI/G/s$	MSE	$T(n)$
$\beta(\alpha_1,\alpha_2)$	GPM	MSER	TGFSR
$B(t, p)$	$h(x)$	MT19937	triang(a, b, m)
$B(\alpha_1,\alpha_2)$	H_0	$N(\mu,\sigma^2)$	$u(n)$
$\mathrm{B}(t)$	H_1	$N(0,1)$	$\hat{u}(n)$
C_{ij}	H&W	$N_d(\boldsymbol{\mu},\Sigma)$	U
C_j	HLA	NC	$U(a, b)$
CCD	IID	negbin(s, p)	$U(0, 1)$
CNI	JSB(α_1,α_2,a,b)	NETA	var()
cor	JSU$(\alpha_1,\alpha_2,\gamma,\beta)$	N&M	VARTA
cov	KN	NORTA	VIP
CPU	KN++	NSGS	VRT
CRN	$l(\theta)$	OCBA	WASSP
cv	L	PMMLOG	Weibull(α, β)
CV	$L(\theta)$	$p(x)$	WELL
d	L&C	$p(x, y)$	$w.p.$
D&D	LCG	$P()$	w
DES	LFC	Pareto(c,α_2)	$w(n)$
$d(n)$	LFSR	$P(\lambda)$	$\hat{w}(n)$
$\hat{d}(n)$	LHD	PT5(α,β)	$\tilde{w}(n)$
df	LIFO	PT6$(\alpha_1,\alpha_2,\beta)$	W_i
D_i	$LL(\alpha,\beta)$	Q	x_q
$DU(i, j)$	$LN(\mu,\sigma^2)$	$q(n)$	$x_{0.5}$
$E()$	$L(t)$	$\hat{q}(n)$	$\boldsymbol{x}$
EAR	m	$Q(t)$	$\boldsymbol{X}$
Erlang	MC	R	$\boldsymbol{X}_k$
expo(β)	MCB	RTI	$X_{(i)}$
FIFO	$M/E_2/1$	SBatch	$\overline{X}_{(n)}$

（续）

符号或缩写	符号或缩写	符号或缩写	符号或缩写
$\overline{Y}_i(w)$	$\Lambda(t)$	Σ	$\sim$
$z_{1-\alpha/2}$	μ	$\hat{\Sigma}$	$\xrightarrow{D}$
α	$\mu, \hat{\mu}$	$\Phi(z)$	$\binom{t}{x}$
β	v	$\chi^2_{k-1,1-\alpha}$	$\lfloor x \rfloor$
γ	ρ	$\Psi(\hat{\alpha})$	$\lceil x \rceil$
$\Gamma(\alpha)$	ρ_{ij}	ω	$\{\}$
ζ	ρ_j	$\hat{}$	
λ	σ	$\approx$	
$\lambda(t)$	σ^2	$\in$	

前 言

本书第5版的目标和前4版仍然保持一致：提供仿真研究的所有重要方面综合的与最新的进展，包括建模、仿真软件、模型校验和确认、输入建模、随机数发生器、随机变量和过程的产生、统计设计和仿真实验的分析，以及强调像制造这样的最重要的应用领域。本书旨在促进读者直观理解仿真与建模，并以技术上正确但更清晰的方式来阐述它们。全书有许多例子和习题，并有广泛的仿真参考文献和著作以供读者进一步研究。

本书可作为各种课程的基本内容，例如：

- 工程、制造、商业或计算机科学专业的高年级大学生或低年级研究生的第一门课（第1～4章以及第5～9章和第13章的一部分）。在结束该课程时，学生要准备进行完整而有效的仿真学习，并进行高级仿真课程的学习。
- 上述任意学科的研究生的第二门仿真课程(第5～12章的大部分)。完成该课程后，学生应该熟悉仿真学习中更高级的方法学问题，也应该准备理解并进行仿真研究。
- 作为运筹学和管理科学中通用课程一部分的仿真导论(第1章、第3章、第5章、第6章、第9章及第13章的一部分)。

对用本书作为教材的教师来说，可以从网址 www.mhhe.com/law 下载各种教学辅助支撑资料。这些资料包括习题答案的全集，第1章、第2章和第7章中的全部仿真模型以及随机数发生器的代码。

本书还可以作为仿真从业者和研究人员的权威参考书。本书末尾对作者本人的诸多实践和咨询项目的实际例子进行了详细讨论。作者还做了巨大努力将主题与相关的研究文献联系起来，既有纸质的，也有在线的，并保持这些材料实时更新。理解本书的前提是有微积分、概率与统计学方面的基础知识(虽然第4章也涉及这些主题)，以及会用计算机做一些实验。对第1章和第2章，读者还应该熟悉通用编程语言，例如C语言。偶尔书中也会用到少量的线性代数或矩阵理论。较深或有技术难度的内容放在标有星号的节中或在各章的附录中。

对第4版的内容进行了大量的修改和补充完成了第5版，但结构绝大部分仍然保持相同。第1章中部分仿真内容移到第13章，留到后面讨论。第2章关于复杂系统建模的内容已经更新，反映了有效事件列表管理的最新研究内容。第3章进行了重写和扩展，反映了仿真软件的最新进展，给出了三个用通用仿真软件包实现的一个通用的仿真案例。第4章讨论了置信区间与假设检验，补充了较多新内容，这一章中概率与统计学方面的基础知识为后面章节提供了基础。第5章对校验和标定做了更为清楚的区别，以避免混淆两者。第6章说明了输入模型不确定性和模型到达过程的最新进展。第7章提供了最有用的随机数发生器。第8章少量更新了关于随机变量和随机过程产生的内容。许多统计设计与分析方法在第9～12章进行扩展和更新，以反映当前的实践水平和近期的研究进展。特别是，第9章综合讨论了用于估计仿真系统的稳态均值的最新方法。更新了第10章中排序与选择程序的讨论以反映其较新的、有效的方法，这些方法并非基于传统的一致区间方法。第11章涉及的方差缩减技术仅有少量改动。第12章讨论了广泛得多且独立的经典实验设计与元模型，还特别强调了哪些设计和元模型专门用于仿真模型。新增的第14章讨论制造系统仿真，内容参见本书的配套网站，而不包含在纸质书里。第14章中关于制造系统仿真应用和最新仿真软件包的内容已经进行了同步更新。本书增加了第13章，讨论了基于Agent的仿真和系统动力学，以及在第4版第1章中提及的其他类型的仿真。学生版的

ExpertFit 分布拟合软件可以在本书配套网站中找到，它可以用来分析第 6 章例子和习题中相关的数据集。为了方便读者阅读和节约篇幅，书后列出了所有参考文献。最后的中英文名词对照增强了本书作为参考文献的价值。

www.CourseSmart.com 网站上有本书的电子版。在该网站中购买电子版图书可以节约购买纸质书的费用，减少环境污染，也可利用网站上的工具进行学习。CourseSmart 电子书可以在线阅读和下载到计算机上。电子书允许读者全文搜索、高亮显示与注释，以及与别人共享注释。CourseSmart 电子书在任何一个地方都可以使用，访问 www.CourseSmart.com 可以了解很多细节和试读样章。

首先应该感谢本人以前的合作者 David Kelton，他对本书的前三版做出了巨大贡献。第 5 版的评审人员给予了我极大的帮助并对初稿给出了深入的反馈意见，从而使本书在内容和表达两个方面都得到了极大的加强。这些评审人员有：Christos Alexopoulos（佐治亚理工学院），Russell Barton（宾夕法尼亚州立大学），Chun-Hung Chen（乔治梅森大学），Shane Henderson（康奈尔大学），Jack Kleijnen（蒂尔堡大学），Pierre L'Ecuyer（蒙特利尔大学），Michael North（美国阿贡国家实验室），以及 Douglas Samuelson（InfoLogix 公司）。众所周知，本书一定会有不少因疏忽而产生的令人遗憾的错误，虽然如此，本人还是感谢以各种方式给予帮助的以下人员：Wayne Adams，Mark Anderson，Sigrun Andradottir，Jay April，Robert Axtell，Emmett Beeker，Marco Better，Edmund Bitinas，A. J. Bobo，Andrei Borshchev，Nathanael Brown，John Carson，Loren Cobb，Eric Frisco，David Galligan，Nigel Gilbert，Fred Glover，David Goldsman，Daniel Green，Charles Harrell，Thomas Hayson，James Henriksen，Raymond Hill，Kathryn Hoad，Terril Hurst，Andrew Hachinski，Jeffrey Joines，Harry King，David Krahl，Emily Lada，Michael Lauren，Steffi Law，Thomas Lucas，Gregory McIntosh，Janet McLeavey，Anup Mokashi，Daniel Muller，Rodney Myers，William Nordgren，Ernie Page，Dennis Pegden，David Peterson，Stuart Eobinson，Paul Sanchez，Susan Sanchez，Lee Schruben，David Siebert，Jeffery Smith，David Sturrock，Ali Tafazzoli，Andrew Waller，Hong Wan，Robert Weber，Preston White，James Wilson。

埃弗里尔 M. 劳
于亚利桑那州图森市

目　　录

第1章 仿真建模入门

1.1 仿真的本质

这是一本有关如何使用计算机对现实世界中各种设备或过程进行模拟或仿真的技术书。所关心的设备或过程通常称为*系统*。为了更科学地研究系统，我们往往需要对其如何运行建立一系列假设，这些假设就构成了*模型*。模型通常以数学公式或逻辑关系的形式表示，力图用于更好地理解相关系统的行为表现如何。

如果组成模型的关系足够简单，则可用数学方法(例如代数、微积分理论或概率论等)对于所关心的问题获得所需的精确信息，称之为*解析解*。然而大多数现实世界中的系统都过于复杂，对实际模型无法使用解析方法进行评估，这些模型必须通过仿真方法进行研究。在*仿真*中，通过计算机对模型进行*数值*评估，通过搜集数据来*估计*该模型的期望真实特征。

作为仿真应用的一个例子，考虑一个制造企业，该企业正考虑扩建一个工厂，但并不确知所获得的产量增加是否能弥补扩建的费用。如果扩建达不到这个要求而将其拆除，则显然是不合算的。然而，深入的仿真研究能解决上面所提到的问题，办法是对该工厂现在的运作以及扩建后的运作进行仿真。

仿真应用的领域是十分广泛的，人们发现，仿真对于下列问题是常用且有效的工具：

- 制造系统的设计与分析；
- 军用武器系统及其后勤保障的评估；
- 通信网络中硬件需求或协议的确定；
- 计算机系统软硬件需求的确定；
- 运输系统(如机场、高速公路、港口、地铁等)的设计与运营；
- 服务组织(如呼叫中心、快餐店、医院、邮局等)的评估设计；
- 业务流程的重组；
- 供应链分析；
- 库存系统订货策略的确定；
- 采矿作业的分析。

仿真是运筹学和管理科学最广泛应用的技术之一，至少在这两个领域，仿真是应用最广泛的。冬季仿真会议就说明了这一点，它每年吸引了600至800人参加。此外，还有其他一些仿真会议，每年有超过100人参加。

还有一些关于运筹学技术应用的调查。例如，Lane，Mansour和Harpell(1993)从一项长期研究中指出，自1973年至1988年，仿真一直被评为三个最重要的“运筹学技术”之一。另外两个技术是数学规划(该术语包括诸如线性规划、非线性规划等许多单项技术)和统计学(它本质上并不是一项运筹学技术)。Gupta(1997)分析了期刊“Interfaces”(有关运筹学应用的引领杂志之一)中从1970年至1992年的1294篇文章，得出在所考虑的13项技术中仿真是仅次于数学规划的一项技术。

但是，仍存在影响人们更广泛接受和应用仿真的几个障碍。首先，用于大规模系统的模型变得非常复杂，编写计算机程序来执行这些模型的确可能是一件烦琐的任务。近年来开发出来的一些优秀的仿真软件产品使得这一任务变得比较容易了，它们具有对仿真模型

编程所需要的许多特点。复杂系统仿真的第二个问题是仿真往往需要耗费大量时间。但是，随着计算机运行速度越来越快，费用越来越低，这一困难也就变得不那么严重了。最后，有一种不幸的印象，虽然仿真复杂一些，但是有人认为仿真只是一个计算机编程练习，因此，为得到“答案”，很多仿真“研究”仅包括启发式模型建立、编程，以及程序的一次运行。我们担心的是，这种态度无疑使得很多仿真研究得出了错误的结论，因为它忽视了如何用一个正确编码模型对所感兴趣的系统进行推断这样一个重要问题。仿真方法学中的这些问题与使用的软硬件很大程度上是独立的，它们构成了在本书后面的章节的一个完整的部分。

有关仿真建模的历史发展的评述可参阅 Nance 和 Sargent(2002)的文献。

在本章的后续内容(以及第 2 章)中，我们将更详细地讨论系统和模型，然后展示如何用通用语言编写计算机程序来仿真不同复杂程度的系统。本章中给出的所有计算机代码均可从 www. mhhe. com/law 下载。

1.2　系统、模型及仿真

系统是实体的集合，例如人或机器，它们为实现某一逻辑目标而动作和相互作用(该定义由 Schmidt 和 Taylor 于 1970 年提出)。在实际应用中，“系统”的含义往往取决于一个特定研究的目标。在某项研究中组成一个系统的实体集合可能是另一项研究中整个系统的一个子集。例如，研究一个银行，确定需要多少出纳员以满足顾客需求。如果仅考虑存取款情况，系统可定义为银行的一部分，仅由服务员以及排队等待或正被服务的顾客组成；如果还要考虑贷款负责人以及保险箱，则系统显然要再进一步扩大[参见 Fishman，(1978，第 3 页)]。我们将系统的状态定义为变量的集合，相对于研究目标，这些变量对描述某一特定时刻的系统是必需的。在研究银行时，忙期出纳员数、银行中的顾客数，以及银行中每个顾客的到达时间是可能的状态变量。

我们将系统归为两种类型：离散的和连续的。离散系统是指状态变量在一些离散时间点上发生瞬时变化的系统。银行就是一个离散系统的例子，因为状态变量(例如，银行中的顾客个数)只有当顾客到达或离开银行时才发生变化。连续系统是指状态变量随时间变化而连续变化的系统。例如，一架在空中飞行的飞机就是一个连续系统的例子，因为如位置、速度等状态变量可随时间变化而连续变化。实际中，很少有系统是完全离散或完全连续的；但是，对大多数系统来说，因为总有一种类型占主导地位，将一个系统分类为连续或离散通常是可能的。

在大多数系统存活期的某些点上，研究系统的要求是尽可能深入了解各个组件之间的关系，或是预测系统在考虑某些新条件时的性能。图 1.1 勾画了研究一个系统的不同方法。

图 1.1　研究系统的方法

● 用实际系统的实验与用系统模型的实验

如果物理上改变系统可能(并且合算)且在新条件下可操作，这样做恐怕是所希望的，因为在这种情况下，有关我们的研究是否有效是没有疑问的。然而，做到这一点是非常少见的，因为这样的实验往往花费巨大或者造成系统破坏。例如，某银行正考虑削减出纳员以减少开支，但是这样一来将会导致顾客等待时间加长甚至流失顾客。更形象地说，这个“系统”甚至可能不存在，然而，无论如何，我们要首先用各种备选配置看看应该如何构建系统，这种情况的例子，如一个建议的通信网络或者一个战略核武器系统。基于这些原

因，通常需要建立一个模型来代表一个系统，并将其作为该实际系统的替代来研究它。当用到一个模型时，总会有一个疑问：该模型是否根据要做的决策精确地反映了系统，这个模型的有效性问题将在第5章中详细介绍。

● 物理模型与数学模型

对大多数人来说，“模型”一词让人想起风洞里面的泥汽车、飞机上拆下来的用于飞行员训练的驾驶座，或者水池中行驶的模拟巨轮模型。这些都是物理模型(或称为形象模型)的例子，并且通常不是运筹学或者系统分析中感兴趣的典型模型。然而，建立物理模型来研究工程或者管理系统有时偶尔也是有用的，如物料储运系统的桌面规模模型，并且至少有一个案例，在仓库中建立一个快餐馆的全尺度物理模型，具有全尺度的、真实(假设饥饿)人群[参见 Swart 与 Donno(1981)]。但是为此目的而建立的绝大多数模型都是数学模型，用逻辑或数量关系来描述一个系统，通过操作或改变这些关系观察模型的反应，从而观察系统是如何反应的——如果数学模型有效的话。也许数学模型的一个最简单的例子就是人们熟知的关系 $d=rt$，其中 r 是移动的速率，t 是移动所花费的时间，d 是移动的距离。这个关系式在一种场景中可能提供了一个有效模型(比如，一个已获得飞行速度的向另一行星驶去的航天探测器)，但对其他目的则可能是非常糟糕的模型(比如在拥挤的城市高速公路的高峰期计算)。

● 解析解与仿真

一旦我们建好一个数学模型，则需要检查它是如何用于回答其假设所代表的系统中我们所关注的问题的。如果这个模型足够简单，则可能通过其关系式和数值得到一个精确的解析解。在 $d=rt$ 这个例子中，如果已知移动距离和速率，则根据模型我们能得到 $t=d/r$ 作为所要求的时间。这是一个非常简单的例子，仅仅通过纸笔就可求得闭式解，但是某些解析解会变得非常复杂，需要大量计算资源。求解一个大的非稀疏矩阵的逆就是这样一个众所周知的例子，虽然有原理上已知的解析公式，但是对具体例子的数值求解却并非易事。如果一个数学模型的解析解是可求的，并且其计算效率是可接受的，则通常直接用解析方法求解模型，而不是用仿真。然而，很多系统相当复杂，从而其有效的数学模型本身很复杂，不可能解析求解。在这种情况下，对模型的研究必须用仿真的方法，即对模型进行数字实验，对其施加输入，观察它们如何影响系统的输出性能。

有时将具有贬义的老锯(老方法)用于描述仿真，像“最后一招”这样的，虽然也许有点对，但事实是在很多情况下，我们一下子就想到要用仿真，因为所感兴趣的系统以及以有效的方式表示系统的模型都是非常复杂的。

进一步，已知用仿真的方法来研究的数学模型(以下我们称之为仿真模型)，我们必须找到特定的工具来完成仿真。为此，可以按三个不同的域来对仿真模型进行分类。

● 静态与动态仿真模型

静态仿真模型是系统在某一特定时刻的表示，或者是可用于表示一个时间不起作用的系统。某些蒙特卡罗模型是静态仿真的例子，这将在第1.8.3小节中讨论。另一方面，动态仿真模型描述随时间变化而不断变化的系统，例如工厂中的传送带系统。

● 确定的与随机的仿真模型

如果一个仿真模型不含任何概率(即随机)成分，则称为确定性的；一个描述某化学反应的微分方程的复杂(且难以解析求解的)系统可能就是这样一个模型。不管它需要花费多少时间来计算以及评价该系统到底如何，在确定性模型中，一旦给定模型的输入数值及关系，其输出结果是“确定的”。然而，很多系统建模都必须至少有一些随机输入成分，这就形成了随机仿真模型(系统建模时忽略随机因素的危险的例子参见第4.7节)。大多数排队和库存系统都是随机模型。随机仿真模型所产生的输出结果本身是随机的，从而只能将其看做模型真实特征的一个估计，这也是仿真的主要缺点之一(参见第1.8节)，这将在本书第9章至第12章中讨论。

● 连续与离散仿真模型

不太严格的讲，我们可用类似之前定义离散和连续系统的方式来定义离散和连续仿真模型。关于离散(事件)仿真和连续仿真的更精确的定义将分别在第 1.3 节和第 13.3 节中给出。值得一提的是，离散模型并不总用于离散系统建模，反之亦然。一个特定系统用离散还是连续模型来描述，这取决于研究的特定目标。例如，高速公路交通流量的模型，如果侧重于各个车辆的特征及运动，则该模型是离散的。反之，如果车辆被看做是一个“整体”，交通流可用连续模型的微分方程描述。更多的相关讨论详见第 5.2 节，尤其是例 5.2。

除了第 13 章外，本书以后要考虑的仿真模型都是离散的、动态的且随机的，统称为离散事件仿真模型(因为确定性模型是随机模型的一个特例，限定随机模型并不失一般性)。

1.3 离散事件仿真

离散事件仿真关注的是这样一类系统的建模，即系统随时间推进，其状态变量是在离散的时间点上瞬时改变的(用更数学化的术语，我们可以说，系统只能在可数时间点上变化)。这些时间点就是事件的发生点，其中事件定义为可能改变系统状态的瞬时发生的行为。尽管离散事件仿真从概念上来说可由手工计算完成，但大多数实际系统必须存储和操作大量数据，这时的离散事件仿真必须由计算机完成(第 1.4.2 小节中我们将举一个小的手工仿真的例子，这只是为了演示所含的逻辑过程)。

例 1.1 考虑一个单服务台的服务设备——例如，只有一个理发师的理发店，或者机场的一个信息服务台——我们将估计(期望的)到达顾客在队(列)中的平均延误时间，此处，一个顾客在队列中的延误时间是指其到达服务设备的时刻与开始接受服务的时刻之间的时间区间长度。为了评估一个顾客的平均延误时间，离散事件仿真模型的状态变量为：服务台的状态(即空闲或者忙碌)、等待队列中的顾客个数(可能为 0)，以及每位等待顾客到达的时刻。服务台的状态需要根据顾客的到达情况来决定，看顾客能立即被服务还是需要排在队尾等待。当服务台完成一个顾客的服务时，队列中的顾客数将决定服务台是空闲还是开始为队首顾客服务。顾客的到达时刻将用于计算顾客在队列中的延误时间，即用顾客开始被服务的时刻(很快将得知)减去他的到达时刻所得的时间差。这个系统有两类事件：顾客的到达，以及服务台完成一个顾客的服务，即顾客的离去。一次到达是一个事件，因为它将导致(状态变量)服务台状态从空闲到忙碌的改变，或者队列中的顾客数减 1。第 1.4 节将详细介绍这个单服务台排队系统是如何建立一个离散事件仿真模型的。◀

在上面这个例子中，两类事件都直接改变了系统状态，但是有些离散事件仿真模型的事件并不用来改变系统状态。例如，某一事件可能用来计划仿真运行结束的时刻(参见第 1.4.6 小节)，而并不能直接导致系统状态的改变。这也是为什么我们之前会谈到了一个事件可能导致系统状态的改变。

1.3.1 时间推进机制

由于离散事件仿真模型的动态本质，我们必须随着仿真的进行保持对仿真时间的当前值的跟踪，同时我们还需要一个将仿真时间从一个值推进到另一个值的机制。我们将仿真模型中给出仿真时间当前值的这个变量称为*仿真钟*。当一个模型用通用语言(如 C 语言)描述时，仿真钟的时间单位从不显式表达，而是假定与输入参数的单位相同。此外，仿真钟通常与计算机仿真运行需要的时间无关。

历史上，有两种主要方法用来推进仿真钟：*下一最早发生事件时间推进*和*固定增量时间推进*。由于所有主流仿真软件和大多数用通用语言编程建模的人都使用第一种方法，并且第二种方法实际上是第一种方法的一个特例，因此本书中讨论的所有离散事件仿真模型

都将采用下一最早发生事件时间推进方法。关于固定增量时间推进的一个简单介绍将在附录1A中给出(本章末)。

对于下一最早发生事件时间推进方法，仿真钟初始化为零，且将来事件的发生时间是确定的。然后仿真钟被推进到这些未来事件中的最早要发生(第一个)事件的发生时刻，此时系统状态被更新以说明一个事件已发生的这一事实，且有关未来事件发生时间的内容也被更新。接下来仿真钟被推进到(新的)下一个最早发生事件的时刻，更新系统状态和确定将来事件时间，依此类推。这种将仿真钟从一个事件时间推进到另一个事件时间的过程不断进行，直至最终满足某一事先设定的终止条件为止。由于所有的状态改变都只发生在离散事件仿真模型的事件时刻，因此，可用将仿真钟从一个事件时间直接跳到下一个事件时间的办法，使非活动时间段被跳过(固定增量时间推进则不跳过这些非活动时间段，这将消耗大量计算机时间，参见附录1A)。值得注意的是，仿真钟连续几次的跳跃区间长度通常是不同的(或不相等的)。

例1.2 下面我们来具体说明例1.1中单服务台排队系统的下一事件时间推进方法。需要用到如下符号：

t_i 为第 i 位顾客的到达时间($t_0=0$)；

$A_i=t_i-t_{i-1}$ 为第 $i-1$ 和第 i 位顾客到达的间隔时间；

S_i 为第 i 位顾客实际花费的服务时间(不包括顾客在队列中等待时间)；

D_i 为第 i 位顾客在队列中的延误时间；

$c_i=t_i+D_i+S_i$ 为第 i 位顾客完成服务离开的时间；

e_i 为任意类型事件的第 i 个发生的时间(仿真钟被激活的第 i 个值，不包括 $e_0=0$)。

以上每个变量通常都被定义为随机变量。假设间隔时间 A_1，A_2，…以及服务时间 S_1，S_2，…的概率分布是已知的，且都有累积分布函数(参见第4.2节)：分别用 F_A 和 F_S 表示(确定 F_A 和 F_S 的方法通常是从所感兴趣的系统中搜集数据，然后按照第6章的方法确定与这些数据吻合的分布)。在 $e_0=0$ 时刻，服务台状态为空闲。由 F_A 产生 A_1(第8章将介绍由一个给定分布生成随机观测值的方法)，并将其加到时间轴的0上，则确定了第一个到达时间 t_1。然后仿真钟从 e_0 推进到下一(第一个)事件发生的时间 $e_1=t_1$(见图1.2，其中曲线箭头表示时间钟的推进)。在时刻 t_1 到达的顾客发现服务台空闲，服务台立即进入服务，因而队列中的延误为 $D_1=0$，同时服务台状态由空闲变为忙碌。该到达的顾客完成服务离开的时间 c_1 由 F_S 所生成的 S_1 并将其加到 t_1 计算得到。最后，第二个到达时间 t_2 由 $t_2=t_1+A_2$ 计算而得，其中 A_2 按照分布 F_A 生成。如果 $t_2<c_1$(见图1.2)，则仿真钟从 e_1 推进到下一事件 $e_2=t_2$(如果 $t_2\geqslant c_1$，则将仿真钟从 e_1 推进到 c_1)。t_2 时刻到达的第二位顾客将发现服务台正忙，因此队列中等待的顾客数由0增加到1，同时记下这位顾客到达的时间；然而，他的服务时间 S_2 并不在此时生成。类似地，第三个到达时间 t_3 由 $t_3=t_2+A_3$ 计算而得。如果 $c_1<t_3$(见图1.2)，仿真钟将从 e_2 推进到下一事件 $e_3=c_1$，即第一位顾客完成服务离开的时间，同时队列中的顾客(时刻 t_2 到达的顾客)开始被服务，其队列中的延误时间 $D_2=c_1-t_2$ 与服务完成时间 $c_2=c_1+S_2$(此时 S_2 根据分布 F_S 生成)，队列中的顾客个数由1减为0。如果 $t_3<c_2$，仿真钟由 e_3 推进到下一事件 $e_4=t_3$。依此类推。这个仿真的最终终止的时间是，比如说，当观测到被延误的顾客总数达到某一规定值。

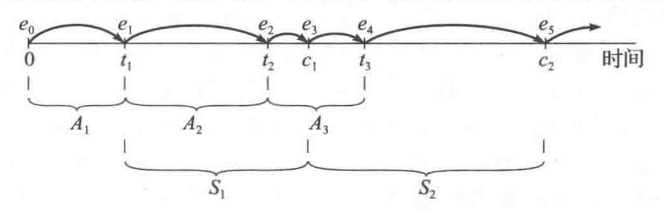


图1.2　单服务台排队系统的下一事件时间推进方法演示

1.3.2 离散事件仿真模型的组件与结构

尽管仿真广泛用于各种现实系统，所有离散事件仿真模型有很多共同组件，并且这些成分有一个逻辑组织，这种组织促进了仿真模型的计算机程序的编程、调试，以及进一步改变。特别是，在采用通用语言编程使用下一事件时间推进方法的大多数离散事件仿真模型中，下面的组件将可以找到。

- *系统状态*：描述在某一特定时刻系统所必需的一组状态变量的集合。
- *仿真钟*：给出仿真时间当前值的变量。
- *事件表*：包含下一次每类事件发生时间的表。
- *统计计数器*：用于存储系统性能统计信息的变量。
- *初始化例程*：零时刻对仿真模型进行初始化的子程序。
- *定时例程*：用于从事件表中确定下一事件并将仿真钟推进到该下一事件发生时间的一个子程序。
- *事件例程*：当某一特定类型事件发生时更新系统状态的子程序(每类事件有一个事件例程)。
- *库例程*：根据确定的概率分布产生随机观测值的一组子程序，作为仿真模型的一部分。
- *报告生成器*：计算性能期望度量的估计值(通过统计计数器)，并在仿真结束时生成一个报告的一个子程序。
- *主程序*：调用定时例程来确定下一事件，然后将控制转移到相应的事件例程以适当更新系统状态的一个子程序。主程序还可以检查终止条件并在终止时调用报告生成器。

这些组件的逻辑关系(控制流)如图 1.3 所示。仿真从零时刻开始，由主程序调用初始化例程，置仿真钟为零，对系统状态、统计计数器，及事件表进行初始化。控制权返回主程序后，它调用定时例程，确定哪类事件最先发生。如果类型 i 的事件是下一个发生的，则仿真钟推进到事件类型 i 将要发生的时刻，并且控制返回到主程序。接下来主程序调用事件例程 i，这里将发生三类典型的活动：(1)更新系统状态，以表明类型 i 的事件已发生；(2)通过更新统计计数器来收集系统性能的信息；(3)生成将来事件的发生时间，并将其添加到事件表中。通常情况下，需要根据概率分布生成随机观测值以确定这些将来事件的发生时间；这个生成的观测值称为*随机变量*。所有过程完成之后，或者由事件例程 i，或者由主程序(根据某终止条件)检验仿真是否应该终止。如果仿真终止条件满足，则主程序调用报告生成器，以计算性能期望度量的估计值并生成报告。若仿真还未终止，则控制权返回主程序，循环重复“主程序→定时例程→主程序→事件例程→终止检验”的过程，直到终止条件最终满足为止。

在结束本节之前，关于系统状态还要说几句。如第 1.2 节中提到的，系统是完好定义的实体集合。实体用数值表征，称为*属性*，这些属性是离散事件仿真模型系统状态的一部分。此外，具有相同属性的实体在表(或*文件*，或*数组*)中往往成为一组。对于每个实体，表中有一个包含了该实体属性的*记录*，这些记录在表中按一定顺序存放，存放顺序取决于某些规定的规则(参见第 2 章对存储记录表的有效方法的讨论)。对于例 1.1 和例 1.2 中的单服务台排队系统，实体是系统中的服务台和顾客。服务台的属性是“服务台状态”(忙或闲)，队列中等待的顾客的属性是“到达时间”(队列中的顾客数也可以视作服务台的一个属性)。此外，正如我们将在第 1.4 节中看到的，队列中的这些顾客将分组在一个表中。

当采用通用语言(如 C 语言)编程仿真时，以上介绍的采用下一事件时间推进机制的离散事件仿真程序的结构和动作是相当典型的，这种方法称为*事件调度法仿真建模*，因为将来事件的时间已明确编码在模型中，这些时间在仿真的将来被调度发生。值得一提的是，另外还有一种仿真建模方法，称作*进程法*，它代之以关注涉及的每个实体的仿真，并编码将一个“典型的”实体的“经历”描述为“流过”系统；用进程观点对仿真建模编程往往

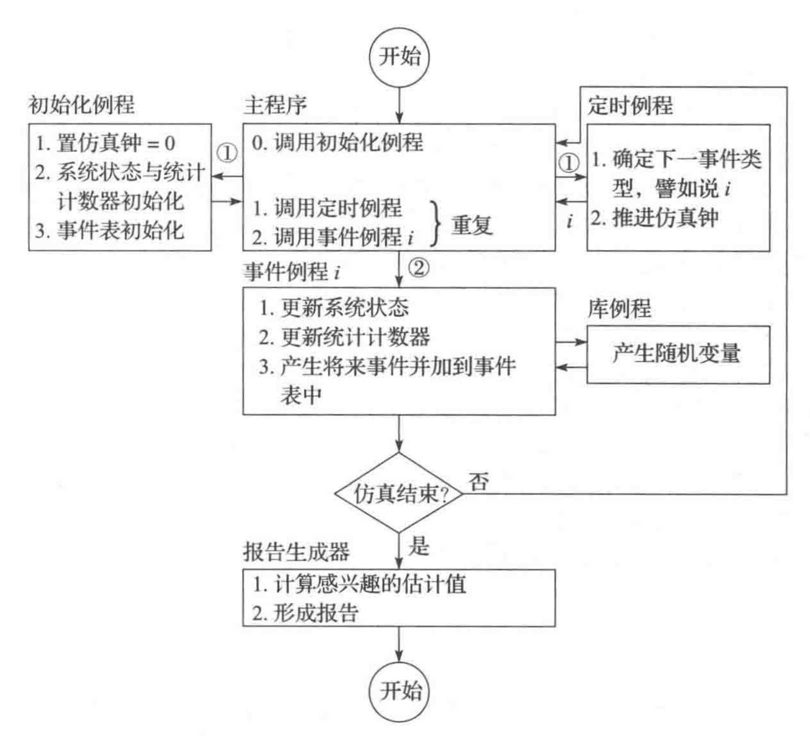


图 1.3　下一事件时间推进方法的控制流

需要用到专用的软件，我们将在第 3 章中讨论。然而，即使采用进程法，仿真实际上在后台仍是按照上面介绍的事件调度逻辑执行的。

1.4　单服务台排队系统的仿真

本章详细说明如何对一个单服务台排队系统进行仿真，如只有一个理发师的理发店。与实际通常关注的系统相比，尽管这个系统看起来非常简单，然而它的仿真相当程度上代表了非常复杂的仿真操作。

在第 1.4.1 小节，我们将更详细地描述所关注的系统，并说明我们的目标；在第 1.4.2 小节，通过展示在每个事件发生后被仿真系统的“快照”，我们直观地解释如何对这个系统进行仿真；第 1.4.3 小节描述了与语言无关的组织结构以及在第 1.4.4 小节中给出的 C 代码的逻辑；仿真结果将在第 1.4.5 小节中讨论，第 1.4.6 小节将终止规则换成一个更常见的方式来终止仿真；最后，第 1.4.7 小节简要介绍一种识别和简化仿真事件与变量结构的技术。

1.4.1　问题描述

考虑这样一个单服务台排队系统(参见图 1.4)，其到达间隔时间 A_1，A_2，…是独立同分布(IID)的随机变量(“同分布”是指到达间隔时间具有相同的概率分布)。顾客到达并发现服务台空闲，则立刻进入服务。顾客的服务时间 S_1，S_2，…也是独立同分布的随机变量，且独立于到达间隔时间。顾客到达并发现服务台忙，则自动加入队尾排队等候。一旦完成一个顾客的服务，服务台根据先进先出(FIFO)原则从队列中选出(如果有)一位顾客(关于其他排队准则以及一般的排队系统的讨论，请参见附录 1B)。



图 1.4　单服务台排队系统

仿真开始于“空且闲”状态，即尚未有顾客出现且服务台闲。在时刻 0，我们将开始等待第一位顾客的到达，即在第一个间隔时间 A_1 之后发生，而不是在时刻 0(时刻 0 到达也可行，但将是另一种不同的建模假设)。我们希望仿真这个系统，直到一定数量(n 个)的顾客完成他们在队列中的延误，即当第 n 个顾客进入服务时仿真结束。需注意的是仿真结束的时间是一个随机变量，取决于到达间隔时间和服务时间随机变量的观测值。

为了要度量这个系统的性能，我们需要寻找三个量的估计值。首先，我们来估计队列中 n 个顾客在仿真中完成各自平均延误时间的期望值，这个量记为 $d(n)$。$d(n)$定义中“期望”一词的意思是：对于一个给定的仿真(或者说，对于一个给定的仿真模型所代表的实际系统的运行)，n 个顾客所观测到的实际平均延误时间依赖于碰巧被观测到的到达间隔时间和服务时间的随机变量观测值。在另一个仿真运行(或者为实际系统的不同时间(天))中，到达时间可能不同，所需要的服务时间也可能不同；这将导致 n 个延误的平均值的不同。因此，一个给定的仿真运行的平均延误时间本身恐怕应视为一个随机变量。我们要估计的 $d(n)$是这个随机变量的期望值。它可以解释为：$d(n)$是许多(实际上，是无数)个(n 个)顾客平均延误时间的平均值。从一次仿真运行中所获得的顾客延误 D_1，D_2，…，D_n，则 $d(n)$的一个显而易见的估计值是：

$$\hat{d}(n)=\frac{\sum_{i=1}^{n}D_i}{n}$$

它就是仿真中观测到的 n 个 D_i的平均值(因此$\hat{d}(n)$也可记为 $\overline{D}(n)$)。本书中，符号上面的符号“^”表示估计值。重要的是要注意，所谓的“延误时间”，我们并不排除其值为零的可能性，即到达顾客发现系统为空或闲时(对本模型，我们知道必有 $D_1=0$)就是这样；在计算平均值时，值为零的延误也要计算进去，因为零延误多表示系统提供了很好的服务，从而我们的输出性能也应体现这一点。用 D_i的平均值而非观察单个 D_i的原因在于，它们的分布未必相同(例如，$D_1=0$，而 D_2 可能大于零)，而且平均值能给出所有顾客延误的单一组合度量。在该意义上说，这并不是通常统计意义上的“平均”，因为每一项并不是同分布下的独立随机观测的。还要注意到，$\hat{d}(n)$本身是基于样本容量为 1 的一个估计，因为我们只进行了一次完整的仿真运行。由基本的统计知识，我们知道，容量为 1 的样本是没有太多价值的；我们将在第 9 章～第 12 章中继续讨论这个问题。

虽然 $d(n)$的估计从顾客角度给出了系统性能的信息，但这样一个系统的管理则可能需要其他信息。事实上，由于大多数实际仿真是相当复杂的，运行起来也非常耗时，我们通常收集性能的很多输出度量值，来描述系统行为的不同方面。例如，考虑队列中(未被服务的)顾客平均数的期望值，记为 $q(n)$，其中 n 是一个必需的标记，它指明了该平均值是在我们的停止规则下观察到 n 个延误这段时间内取的。这个“平均”和队列中延误时间的平均不同，因为它是通过除以时间(连续的)，而非顾客数(离散的)来得到的。因此，我们需要定义什么是队列中顾客数的时间平均值。为此，设 $Q(t)$为 t 时刻队列中的顾客数，$t\geqslant 0$；设 $T(n)$为观察 n 次延误所需要的时间。那么，在 $0\sim T(n)$的任意时刻 t，$Q(t)$是一个非负整数。另外，设 p_i为$Q(t)$等于 i 的时间的期望比例(取值介于 0 和 1 之间)，则 $q(n)$可以合理地定义为：

$$q(n)=\sum_{i=0}^{+\infty}ip_i$$

因此，$q(n)$是一个对队长 $Q(t)$为 i 的加权平均值，其权值为期望比例。为了估计一个仿真中的 $q(n)$，我们只需将 p_i换成其估计，从而得到：

$$\hat{q}(n)=\sum_{i=0}^{+\infty}i\,\hat{p}_i \tag{1.1}$$

其中，$\hat{p}_i$ 为在仿真过程中观测到顾客数为 i 时所占的时间比例。

然而，计算时用几何平均来表示$\hat{q}(n)$会更简洁。设T_i为仿真中队长为i的总时间，则$T(n)=T_0+T_1+T_2+\cdots$且$\hat{p}_i=T_i/T(n)$，则式(1.1)可改写为：

$$\hat{q}(n)=\frac{\sum_{i=0}^{+\infty} iT_i}{T(n)} \tag{1.2}$$

图 1.5 给出了$Q(t)$的一个可能的时间路径，或叫实现，其中，$n=6$；暂时先不考虑阴影部分。到达事件发生在时刻 0.4，1.6，2.1，3.8，4.0，5.6，5.8 和 7.2。离去事件(服务完成)发生在时刻 2.4，3.1，3.3，4.9 和 8.6，且仿真结束在$T(6)=8.6$。

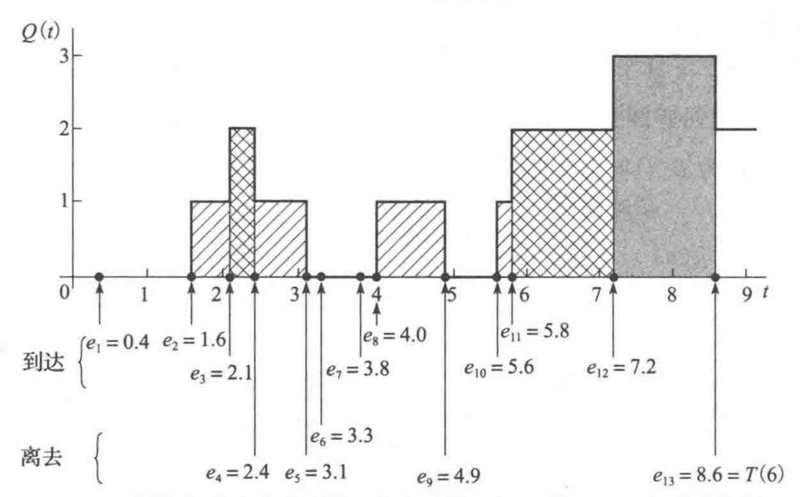


图 1.5　单服务台排队系统的一个实现的$Q(t)$、到达时间和离去时间

注意，在图 1.5 中，$Q(t)$并不记录服务中的顾客个数(如果有的话)，因此在时间段 0.4～1.6 中，系统中有一个顾客正在被服务，但队列是空的(即$Q(t)=0$)；同样的情形发生在时间段 3.1～3.3、3.8～4.0，以及 4.9～5.6 中。然而在时间段 3.3～3.8 中，系统为空且服务台为闲状态，如同时间段 0～0.4 一样。为了计算$\hat{q}(n)$，我们需要首先计算T_i，可以计算图 1.5 中$Q(t)=0$，1，2，…等等的时间段(有时是分开的)得到：

$$T_0=(1.6-0.0)+(4.0-3.1)+(5.6-4.9)=3.2$$
$$T_1=(2.1-1.6)+(3.1-2.4)+(4.9-4.0)+(5.8-5.6)=2.3$$
$$T_2=(2.4-2.1)+(7.2-5.8)=1.7$$
$$T_3=(8.6-7.2)=1.4$$

对于$i\geqslant 4$，$T_i=0$，因为在本次实现中，队长从未超过 3。由式(1.2)得：

$$\sum_{i=0}^{+\infty} iT_i=(0\times 3.2)+(1\times 2.3)+(2\times 1.7)+(3\times 1.4)=9.9 \tag{1.3}$$

因此，这次特定的仿真运行的队列中的时间平均值为$\hat{q}(6)=9.9/8.6=1.15$。下面，注意式(1.3)右端的每个非零项对应图 1.5 所示的一块阴影部分：1×2.3为斜线阴影部分(共四块)，2×1.7是交叉线阴影部分(共两块)，3×1.4为黑阴影部分(一块)。换言之，式(1.2)的数值求和就是从仿真开始到结束的$Q(t)$曲线下的面积。既然“曲线下的面积”是可积的，那么我们可写成：

$$\sum_{i=0}^{+\infty} iT_i=\int_0^{T(n)} Q(t)\,\mathrm{d}t$$

因此，$q(n)$的估计就可以表达为：

$$\hat{q}(n)=\frac{\int_0^{T(n)} Q(t)\,\mathrm{d}t}{T(n)} \tag{1.4}$$

尽管$\hat{q}(n)$的表达式式(1.2)和式(1.4)是等价的，然而我们更倾向于使用式(1.4)，式中的积分可以随着仿真的推进通过计算矩形面积并简单累加而得。当然，这样做没有式(1.2)清晰地进行求和计算那么方便。此外，式(1.4)给出了$Q(t)$的一个连续平均，因为积分可以粗略地看做一个连续求和。

这个系统的第三个也是最后一个输出性能是用来度量服务台忙碌程度的。服务台的期望利用率为仿真期(从0时刻到$T(n)$时刻)服务台忙碌(非空闲)的时间期望比例，因此为一个介于0和1之间的数值，记为$u(n)$。那么，由一个单一仿真，$u(n)$的估计$\hat{u}(n)$=观测到的忙碌时间比例。那么，$\hat{u}(n)$可以在仿真中直接计算：通过记录服务台改变状态(闲到忙或反之)的时间，再进行适当的减法和除法求得。然而，类似于平均队长，将其看做一个连续时间平均将会容易一些，定义“忙函数”为：

$$B(t)=\begin{cases}1, & \text{如果服务器在时刻 } t \text{ 忙碌}\\ 0, & \text{如果服务器在时刻 } t \text{ 空闲}\end{cases}$$

因此$\hat{u}(n)$可以表达为$B(t)$等于1时的时间比例。图1.6描绘了与图1.5所示同一个仿真中的$Q(t)$相对应的$B(t)$。在此情形下，我们得到：

$$\hat{u}(n)=\frac{(3.3-0.4)+(8.6-3.8)}{8.6}=\frac{7.7}{8.6}=0.90 \tag{1.5}$$

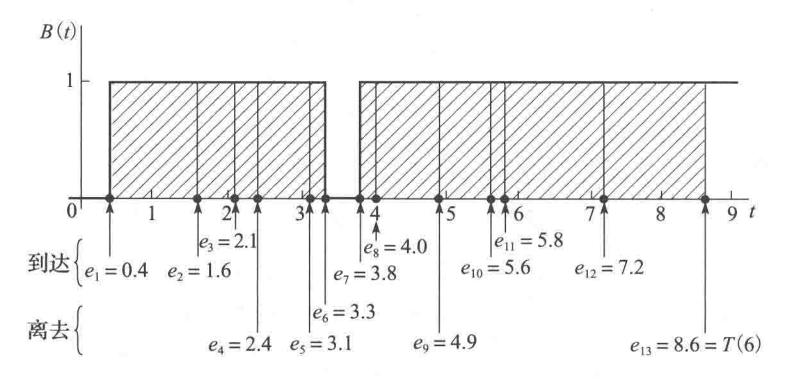


图1.6　单服务台排队系统的一个实现的$B(t)$、到达及离去时间(与图1.5所示的实现相同)

这说明在仿真过程中服务台90%的时间处于忙碌。然而，式(1.5)的数值计算也可看做仿真过程的$B(t)$函数的面积，因为$B(t)$的高度总是为0或1，因此有：

$$\hat{u}(n)=\frac{\int_0^{T(n)}B(t)\,\mathrm{d}t}{T(n)} \tag{1.6}$$

同时我们可以看到，根据前面我们的利用率的解释，$\hat{u}(n)$是$B(t)$函数的连续平均，类似于$\hat{q}(n)$。

将$\hat{u}(n)$写成式(1.6)的积分形式的原因是出于计算的考虑，随着仿真推进，$B(t)$的积分易于通过累加矩形面积获得。对涉及某类“服务台”的很多仿真来说，利用率统计是识别瓶颈(利用率接近100%，加上进入队列的高拥挤的度量)或剩余能力(低利用率)的相当好的办法；对于成本很高的“服务台”，比如生产系统中的机器人或数据处理系统的大型主机，这一点尤其正确。

回顾本节，性能的三个度量是：队列中的平均延误$\hat{d}(n)$、队列中时间平均顾客数$\hat{q}(n)$，以及服务台忙的时间比例$\hat{u}(n)$。队列中的平均延误是离散时间统计的一个例子，因为它相对于一组随机变量$\{D_i\}$(有着离散的“时间”下标$i=1, 2, \cdots$)来定义的。队列中的时间平均顾客数和服务台忙的时间比例都是连续时间统计的例子，因为它们的定义分别基于一组随机变量$\{Q(t)\}$和$\{B(t)\}$——都有连续时间参数$t\in[0, +\infty)$(符号“$\in$”的意思是“属于”，表示t可为任意非负实数)。离散时间统计和连续时间统计在仿真中都很常见，也可以用在非平

均统计的场合。例如，我们可能对所观测队列的所有延误的最大值(离散时间统计)感兴趣，或者仿真时至少包含5个顾客的队列的时间比例(连续时间统计)。

本系统的事件为顾客的到达和顾客的离去(服务完成之后)；估计 $d(n)$、$q(n)$ 和 $u(n)$ 所必需的状态变量为：服务台的状态(0为闲；1为忙)，队列中的顾客数，当前队列中每个顾客的到达时间(表示为一个表)，以及最后一个事件(最近的事件)的时间。最后一个事件的时间定义为 e_{i-1}(其中，$e_{i-1} \leqslant t < e_i$，$t$ 为当前仿真时间)，用于在求 $q(n)$ 和 $u(n)$ 的估计时为累加面积计算矩形的宽度。

1.4.2 直观解释

下面我们开始说明如何对一个单服务台排队系统进行仿真，方法是展示在 $e_0=0$，e_1，e_2，…，e_{13} 时刻仿真模型在计算机中是如何表示的，在这13个事件相继发生的时刻，需要观测所要求的队列中的延误顾客数 $n=6$。为方便表示，假设顾客的到达间隔时间和服务时间分别为：

$$A_1 = 0.4, A_2 = 1.2, A_3 = 0.5, A_4 = 1.7, A_5 = 0.2$$
$$A_6 = 1.6, A_7 = 0.2, A_8 = 1.4, A_9 = 1.9, \cdots$$
$$S_1 = 2.0, S_2 = 0.7, S_3 = 0.2, S_4 = 1.1, S_5 = 3.7, S_6 = 0.6, \cdots$$

因此，在时间0和首次到达事件发生时间之间有0.4个时间单位，在第一个和第二个顾客到达之间有1.2个时间单位，依此类推；第一个顾客的服务时间为2.0，等等。这里没必要指明时间单位到底是什么(分钟、小时等)，但须保证所有的时间采用相同的单位。在实际仿真中(参见第1.4.4小节)，仿真执行时，A_i 和 S_i 将根据需要由他们各自的概率分布产生。上述 A_i 和 S_i 的数值为人工设定的，以便产生图1.5和1.6所表示的仿真实现，图中表示了 $Q(t)$ 和 $B(t)$ 的进程。

图1.7给出了在 $e_0=0$，$e_1=0.4$，…，$e_{13}=8.6$ 每个时刻系统本身以及该系统的计算机表示的快照。在"系统"图部分，正方形表示服务台，圆圈表示顾客；顾客圆圈中的数字表示他们到达的时间。在"计算机表示"图部分，所示变量的值是该事件在所有处理都完成之后的值。我们的讨论将集中在不同事件时刻计算机表示是如何变化的。

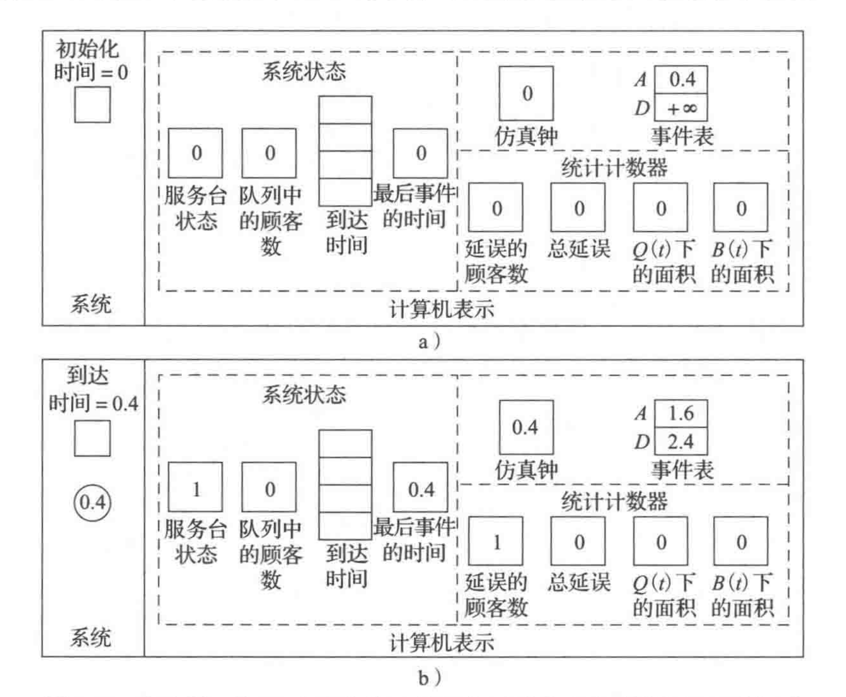
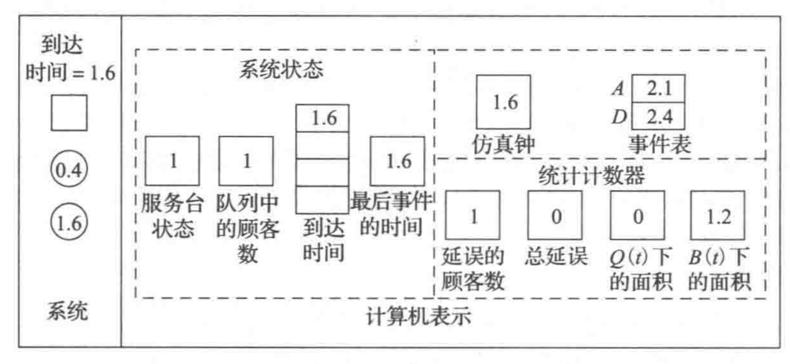
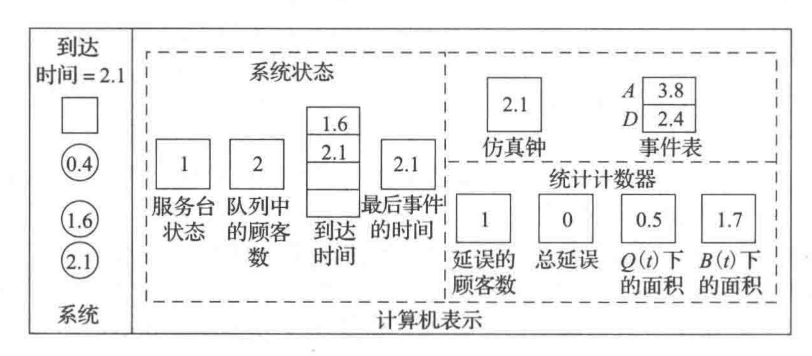


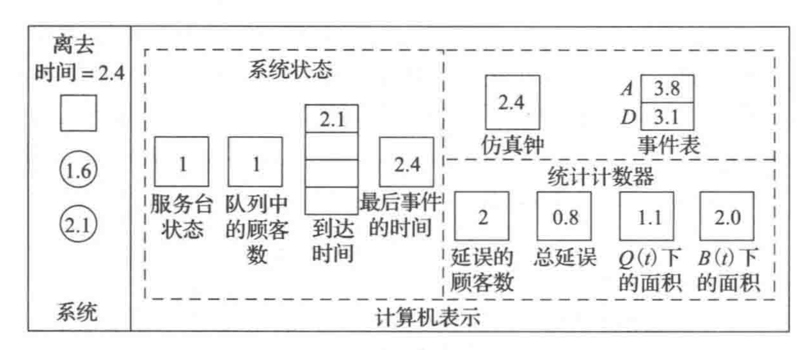
图1.7 在时刻0及13个事件发生的每个时刻系统及其计算机表示快照


到达
时间＝1.6
0.4
1.6
系统
系统状态
1
服务台状态
1
队列中的顾客数
1.6
到达时间
1.6
最后事件的时间
1.6
仿真钟
A 2.1
D 2.4
事件表
统计计数器
1
延误的顾客数
0
总延误
0
Q(t)下的面积
1.2
B(t)下的面积
计算机表示

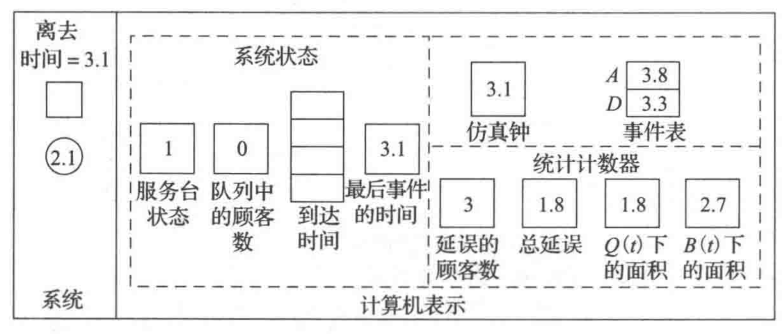
c）


到达
时间＝2.1
0.4
1.6
2.1
系统
系统状态
1
服务台状态
2
队列中的顾客数
1.6
2.1
到达时间
2.1
最后事件的时间
2.1
仿真钟
A 3.8
D 2.4
事件表
统计计数器
1
延误的顾客数
0
总延误
0.5
Q(t)下的面积
1.7
B(t)下的面积
计算机表示

d）

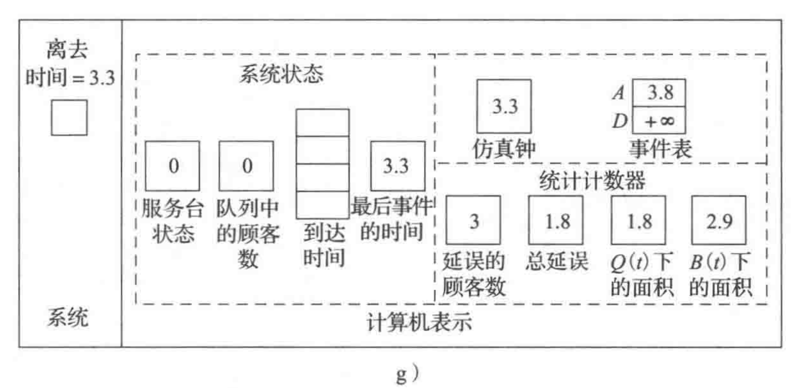

离去
时间＝2.4
1.6
2.1
系统
系统状态
1
服务台状态
1
队列中的顾客数
2.1
到达时间
2.4
最后事件的时间
2.4
仿真钟
A 3.8
D 3.1
事件表
统计计数器
2
延误的顾客数
0.8
总延误
1.1
Q(t)下的面积
2.0
B(t)下的面积
计算机表示

e）

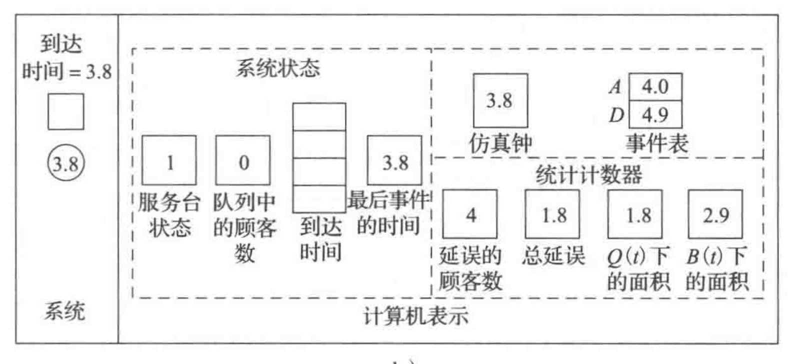

离去
时间＝3.1
2.1
系统
系统状态
1
服务台状态
0
队列中的顾客数
到达时间
3.1
最后事件的时间
3.1
仿真钟
A 3.8
D 3.3
事件表
统计计数器
3
延误的顾客数
1.8
总延误
1.8
Q(t)下的面积
2.7
B(t)下的面积
计算机表示

f）

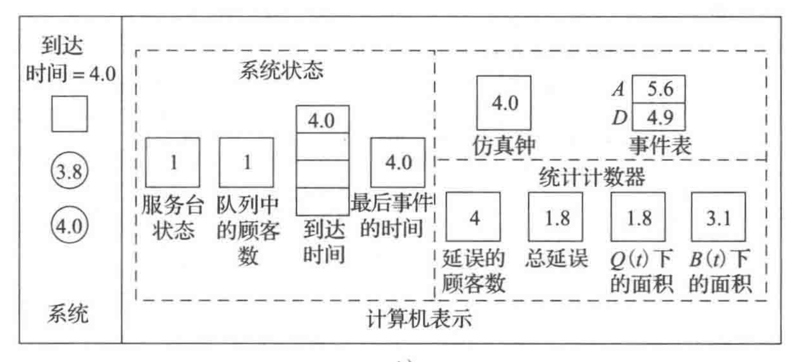
图 1.7 （续）


离去
时间=3.3
系统
系统状态
0
服务台状态
0
队列中的顾客数
到达时间
3.3
最后事件的时间
3.3
仿真钟
A 3.8
D +∞
事件表
统计计数器
3
延误的顾客数
1.8
总延误
1.8
Q(t)下的面积
2.9
B(t)下的面积
计算机表示

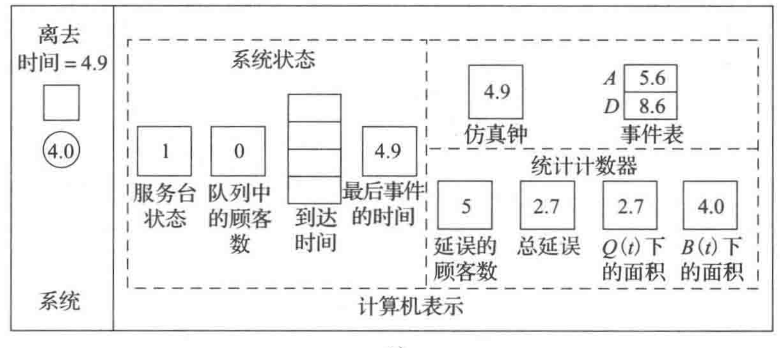
g）


到达
时间=3.8
3.8
系统
系统状态
1
服务台状态
0
队列中的顾客数
到达时间
3.8
最后事件的时间
3.8
仿真钟
A 4.0
D 4.9
事件表
统计计数器
4
延误的顾客数
1.8
总延误
1.8
Q(t)下的面积
2.9
B(t)下的面积
计算机表示

h）


到达
时间=4.0
3.8
4.0
系统
系统状态
1
服务台状态
1
队列中的顾客数
4.0
到达时间
4.0
最后事件的时间
4.0
仿真钟
A 5.6
D 4.9
事件表
统计计数器
4
延误的顾客数
1.8
总延误
1.8
Q(t)下的面积
3.1
B(t)下的面积
计算机表示

i）


离去
时间=4.9
4.0
系统
系统状态
1
服务台状态
0
队列中的顾客数
到达时间
4.9
最后事件的时间
4.9
仿真钟
A 5.6
D 8.6
事件表
统计计数器
5
延误的顾客数
2.7
总延误
2.7
Q(t)下的面积
4.0
B(t)下的面积
计算机表示

j）

图 1.7 （续）

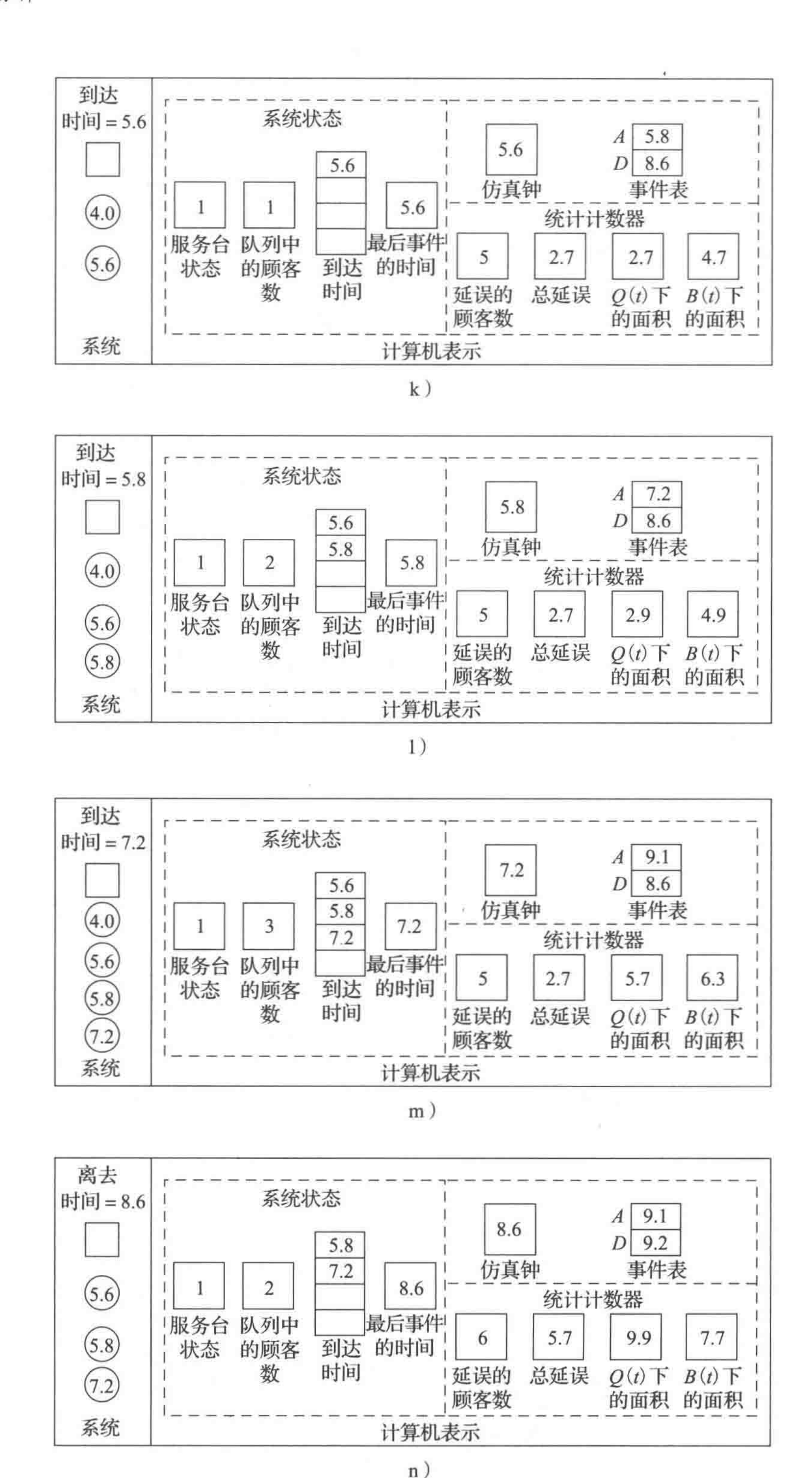


图 1.7 （续）

$t=0$：*初始化* 仿真从主程序调用初始化程序开始。我们的建模假设是系统中没有顾客且服务台闲，如图 1.7a 所示。初始化模型状态变量：服务器状态为 0(0 表示闲，1 表示

忙，类似于 $B(t)$ 函数的定义），队列中的顾客数为 0。用一维数组存储当前队列中顾客的到达时间；这个数组初始化为空，随着仿真推进，其长度可能增加或变短。最后（最近）事件的发生时间初始化为 0，这样在第一个事件发生时刻（此时需要使用它），它才会有一个正确值。仿真钟设为 0，事件表给出接下来发生的每类事件发生的时间，其初始化过程如下。第一个到达事件的发生时间为 $0+A_1=0.4$，用“A”表示到达事件表的下一事件。由于还没有顾客被服务，讨论下一个离去的时间（在事件列表中记为“D”）是没有意义的。至此已知第一个事件为时刻 0.4 时第一位顾客的到达。然而，通常的仿真过程是通过观察事件表，从中选择最小数值来决定下一个将要发生的事件的过程，因此安排下一个离去在时间 $+\infty$ 发生（在计算机中用一个非常大的数值表示），这样我们有效地去除了离去事件的考虑而强制下一事件为一个到达事件。最后，四个统计计数器初始化为 0。当所有初始化完成之后，控制返回到主程序，然后调用定时例程以确定下一事件。由于 $0.4<+\infty$，因此下一事件将在 0.4 时刻到达，定时例程将仿真钟推进到该时刻，然后将控制返回到主程序，并带回下一事件是到达的信息。

$t=0.4$：*顾客 1 的到达*　在时刻 0.4，主程序将控制交给到达例程以处理第一个顾客的到达。图 1.7b 表示了在完成这次到达的处理所引起的所有变化之后的系统和计算机表示。由于该顾客到达时服务台闲（状态为 0），因此立即开始服务，队列中的延误 $D_1=0$（作为一个延误进行统计）。服务台状态设为 1，表示当前为忙状态，但是队列仍为空。仿真钟显示当前时刻 0.4，事件表更新以反映该顾客的到达：下一次到达将在 $A_2=1.2$ 个时间单位之后发生，即 $0.4+1.2=1.6$；下一离去时间（当前到达顾客的服务完成时间）为 $S_1=2.0$ 个时间单位之后，即 $0.4+2.0=2.4$。延误数增加为 1（当其达到 $n=6$ 时仿真结束），$D_1=0$ 加到总延误时间上（总延误时间仍为 0）。更新 $Q(t)$ 曲线下的面积：用上一个 $Q(t)$ 的值（该值从上一事件一直延续到现在，在这里即 0）乘以上一事件到现在的时间差 $t-$（上一事件的时间）$=0.4-0=0.4$，这个乘积加到原面积上。注意，这里所用到的上一事件的时间是指那个老值（0），而不是待更新的新值（0.4）。类似地，更新 $B(t)$ 曲线下的面积：加入上一个值（0）与时间差的乘积（参见图 1.5 和图 1.6）。最后，更新上一事件发生时间为当前时间 0.4，返回主程序。主程序调用定时例程，定时例程扫描事件表，查找最小值，得到下一事件为在时刻 1.6 新到达的事件；更新仿真钟为这个值，返回主程序，并带回下一事件是到达的信息。

$t=1.6$：*顾客 2 的到达*　此时，我们再次进入到达例程，图 1.7c 显示了在处理完这次到达所引起的所有变化之后的系统和计算机表示。由于该顾客到达时服务台忙（在他到达时服务台状态为 1），因此他必须进入队列，在队首位置等待。其到达时间存入数组第一个下标内，队列中顾客数变量升为 1。事件表中的下一个到达时间为 $A_3=0.5$ 个时间单位之后，即更新为 $1.6+0.5=2.1$；下一次离去的时间不变，因为其值为第一个顾客（仍在服务中）的离开时间 2.4。由于当前队列中没有人被延误，因此延误数和总延误变量都不变。$Q(t)$ 所围面积增加量等于 0（前一个 $Q(t)$ 的值）乘以上一事件到现在的时间差 $1.6-0.4=1.2$ 的乘积。$B(t)$ 下的面积增加量等于 1（在前一个 $B(t)$ 值的基础上）乘以相同的时间差 1.2。在完成从上一事件到当前时间的更新后，控制返回到主程序；然后进入定时例程，确定下一事件为在 2.1 时刻的到达事件。

$t=2.1$：*顾客 3 的到达*　再次调用到达例程，如图 1.7d 所示。服务台仍保持忙碌，队列中增加一个顾客，其到达时间存入队列数组的第二个位置。下一到达时间更新为 $t+A_4=2.1+1.7=3.8$，下一次离去时间仍不变，因为服务台仍未完成顾客 1 的服务。延误计数器不变，因为队列中无任何顾客延误结束。两个面积累加器要更新，均增加 1（$Q(t)$ 和 $B(t)$ 的前一个值均为 1）乘以上一事件以来的时间 $2.1-1.6=0.5$。得到上一事件到当前的时间后，返回到主程序，并调用定时例程，搜索事件表以确定下一事件为在时刻 2.4 离去，并将仿真钟修改为该时刻。

$t=2.4$：顾客 1 的离去　这时主程序调用离去例程，如图 1.7e 所示。服务台仍保持忙碌状态，因为顾客 2 从队列中的第一个位置出来进入服务。队列长度减小 1，到达时间数组整体向上移动一个位置，此时顾客 3 在队首。顾客 2 进入服务，需要 $S_2=0.7$ 个时间单位，因此事件表中的下一个离去时间更新为(顾客 2 的)再过 S_2 之后，即 $2.4+0.7=3.1$；下一个到达时间(顾客 4 的)不变，因为这是由顾客 3 的到达早就决定的，而我们仍在等待顾客 4 的到达时间。由于顾客 2 进入服务，已完成他在队列中的等待，因此需要更新延误统计信息。利用到达时间数组，用当前时间减去顾客 2 的到达时间可得其延误时间，即 $D_2=2.4-1.6=0.8$(注意这个值 1.6 是在到达时间数组改变之前就存于其第一位的，因此这个延误计算必须在到达时间数组改变之前完成)。更新两部分的面积：$Q(t)$ 增加 $2\times(2.4-2.1)$(注意需要用到前一个 $Q(t)$ 的值)，$B(t)$ 增加 $1\times(2.4-2.1)$。更新上一事件时间，返回主程序，定时例程确定下一事件为离去事件，时间为 3.1。

$t=3.1$：顾客 2 的离去　该离去的变化类似于上面讨论的 $t=2.4$ 时顾客 1 的离去事件。注意我们观测到队列中另一个顾客的延误，且该事件处理后队列又将变空，但服务器仍为忙。

$t=3.3$：顾客 3 的离开　同样，这里的变化类似于上面两个离去事件的讨论，只有一点例外：由于队列现在为空，服务台为闲，因此我们必须在事件表中将下一离去时间设置为 $+\infty$，因为此时系统和时刻 0 时的状态一样，我们要强制设置下一事件为顾客 4 的到达。

$t=3.8$：顾客 4 的到达　因为该顾客到达时发现服务台空闲，其延误为 0(即 $D_4=0$)，直接进入服务。这样，这里的变化与 $t=0.4$ 时第一个顾客的到达非常类似。

其余的六个事件时间如图 1.7i～图 1.7n 所示，读者可以研究这些图，确信理解了为什么这些变量和数组的变化如图 1.7 所示的那样；顺着图 1.5 和图 1.6 中的 $Q(t)$ 和 $B(t)$ 分析对我们的理解可能是有帮助的。当顾客 5 在时刻 $t=8.6$ 时离去，顾客 6 离开队列进入服务，此时延误的顾客数达到 6(n 的规定值)，仿真结束。主程序调用报告生成器以计算最终输出度量($\hat{d}(6)=5.7/6=0.95$，$\hat{q}(6)=9.9/8.6=1.15$，$\hat{u}(6)=7.7/8.6=0.90$)，并显示出来。

上述例子说明了仿真的逻辑，下面给出几点特别的说明。

- 也许仿真的动态性的关键因素是仿真钟与事件表之间的交互效用。因为通过在每个事件的处理的结束点扫描事件表以得到最小事件时间(即下一事件)，从而事件表得到维护，仿真钟跳到下一个事件。这就是仿真如何随着时间的变化而推进的。
- 当处理一个事件时，"仿真"时间不动。然而，尽管此时时间对模型来说是静止的，仍需注意按照适当的顺序处理状态变量和统计计数器的更新。例如，一个错误的做法是：先更新队列中的顾客数，然后更新 $Q(t)$ 曲线下的面积——求面积所用的矩形的高应该是前一个 $Q(t)$ 的值(在当前事件影响 $Q(t)$ 之前的那个值)。类似地，在更新面积计数器之前更新上一事件的时间也是错误的。然而还有一种错误是，在处理离去事件时，如果一个队列表在计算队首顾客的延误之前就被改变了，则其到达系统的时间将丢失。
- 有时我们容易忽略一些看似例外实则需要处理的地方。例如，人们总容易忽视，在一个顾客离开后队列可能为空，迫使服务台置于闲，下一个离去事件却没有考虑。另外，终止条件也容易引起混淆；在上例中，仿真结束于一个看似"正常"的方式，在一个顾客离去后，允许另一个顾客进入服务，将最后所需要的延误考虑进去，而仿真可能实际上已经结束了，而不是一个到达事件——如何做到的呢？
- 在某些仿真中，可能出现事件表中的两(或以上)项都是最小的，此时需要一个综合的决策规则以解开这种时间结(参见第 1.5 节中的库存仿真)。这个"解结"规则会影响仿真结果，因此其选择必须与系统如何建模保持一致。然而在很多仿真中，我们可以忽略时间结的可能性，因为连续型随机变量的使用使得它们的出现成为一个概率为 0 的事件。例如在上述模型中，如果到达间隔时间或者服务时间的分布是连续的，则事件表中的时间结是一个概率为零的事件(虽然由于实数的表示是有限精度，它在计算机仿真中仍可能发生)。

上述练习是为了说明从事件调度的观点进行离散事件仿真有关变化和数据结构，其中包含了这种类型的更复杂的仿真所需要大部分重要思想。所用到的到达间隔时间和服务时间可以由某种随机数表得到，构建该表以反映所要求的概率分布。这样得到称为手工仿真的结果，原理上可以达到任意长度。这样做显然很枯燥，因此接下来我们将转向用计算机在更长更复杂的仿真中执行有关算法和簿记工作。

1.4.3 程序组织与逻辑

在本节中，我们给出C程序所需要的组件，以便对单服务台排队系统进行仿真，C程序在第1.4.4小节中给出。

这里选择像C语言这样的通用语言而不是更强大的高级仿真软件来介绍计算机仿真，有如下几方面原因。

- 通过学习用通用语言仿真(学习者必须关注每一个细节)，学习者将对仿真如何实际运行有一个更好的了解，从而使高级仿真软件的一个开关晚开而出现概念错误的情况会减少。
- 尽管事实上现在已经有非常好的且功能强大的仿真软件可用(参见第3章)，如果复杂系统中特殊的、细微的逻辑要如实地表示出来的话，有时至少仍然需要用通用语言来编写部分复杂系统的程序。
- 通用语言运用广泛，整个仿真有时仍然用这种方式。

本书的目的不是详细地教授任何特殊的仿真软件，虽然我们在第3章概述了几个软件包。通过学习我们给出的更通用的方法，以及在本章和下章学习我们的仿真，读者会发现学习专用仿真软件产品是比较容易的。

在下一节我们进行仿真的单服务台排队模型与上一节所使用的模型有如下两个方面不同。

- 仿真将结束在队列中完成 $n=1\,000$ 次延误的时刻，而不是之前的 $n=6$。这样做的目的是收集更多的数据(也许是为了给读者展示计算机的耐性，因为我们刚刚在前一节缓慢地手算出了 $n=6$ 的情况)。重要的是，要注意该终止规则的变化改变了模型本身，其中输出度量是相对于终止规则来定义的；因此，被估计的 $d(n)$，$q(n)$和 $u(n)$这些量中出现了符号“n”。
- 到达间隔时间和服务时间现在建模为指数分布的独立随机变量，到达间隔时间的均值为1分钟，服务时间的均值为0.5分钟。均值为 β(任意正实数)的指数分布是连续的，其概率密度函数为：

  $$f(x)=\frac{1}{\beta}\mathrm{e}^{-x/\beta},\quad \forall x\geqslant 0$$

  一般密度函数，特别是指数分布的更多信息，请见第4章和第6章。我们做这种改变的原因是这样产生输入量(它驱动仿真)更常见。比如到达间隔和服务时间，是根据规定的分布产生的，而不是像前一节中我们做的那样，是“已知”的。选取具有上述特定 β 值的指数分布实质上是任意的并且是初定的，因为在计算机上很容易产生指数随机变量(实际上，假设到达间隔时间为指数分布是非常合适的；但是，假设服务时间为指数分布则不是很合适)。对仿真中的输入随机变量建模来说，如何确定分布形式和参数，第6章详细讨论这一重要问题。

具有指数分布的到达间隔和服务时间的单服务台队列一般叫做 $M/M/1$ 队列，在附录1B中有讨论。

为了仿真此模型，我们需要找到根据指数分布生成随机变量的方法：首先，调用一个随机数发生器(在第7章会详述)以生成在0和1之间均匀分布的变量 U；这个分布以后称为 $U(0,1)$，它的密度函数为：

$$f(x)=\begin{cases}1, & 0\leqslant 1\leqslant 1\\ 0, & \text{其他}\end{cases}$$

很容易看出，$U(0,1)$落在[0，1]上任意子区间[x，$x+\Delta x$]内的概率(均匀地)为 Δx

(见第 6.2.2 小节)。$U(0,1)$分布是仿真建模的基础，因为，如同我们将在第 8 章见到的那样，首先生成的一个或者多个$U(0,1)$随机变量，再进行某种变换可以得到任意分布的随机变量。在得到U以后，我们会对其取自然对数，将结果与β相乘，最后改变符号，返回$-\beta\ln U$，我们将证明这是一个均值为β的指数随机变量。

为了了解此算法为何这样做，请回顾一下，随机变量X的(累积)分布函数的定义，对于任意实数x，$F(x)=P(X\leqslant x)$(第 4 章有概率论基础的回顾)。如果X是均值为β的指数分布，那么对于任意实数$x\geqslant 0$，都有：

$$F(x)=\int_0^x \frac{1}{\beta}\mathrm{e}^{-t/\beta}\mathrm{d}t = 1-\mathrm{e}^{-x/\beta}$$

因为对于自变量$t\geqslant 0$，指数分布的概率密度函数是$(1/\beta)\mathrm{e}^{-t/\beta}$。为了说明我们的方法是正确的，我们可以试图验证其返回值小于或等于x(任意非负实数)的概率是上面给出的$F(x)$：

$$\begin{aligned}P(-\beta\ln U\leqslant x) &= P\left(\ln U\geqslant -\frac{x}{\beta}\right)=P(U\geqslant \mathrm{e}^{-x/\beta})\\ &= P(\mathrm{e}^{-x/\beta}\leqslant U\leqslant 1)=1-\mathrm{e}^{-x/\beta}\end{aligned}$$

上面的第一行是由除以$-\beta$得到的(由于$\beta>0$，所以$-\beta<0$，不等式变号)，对两边取指数函数得到第二行(指数函数是单调递增的，所以不等式符号不变)，因为已知U是$U(0,1)$，区间$[\mathrm{e}^{-x/\beta},1]$被包含在区间$[0,1]$中，第三行只是重写。因为最后一行是指数分布的$F(x)$，我们成功验证了我们的算法是正确的。第 8 章一般性地讨论如何生成随机变量及其过程。

在我们的程序里，为了得到上述的变量U，我们将使用特殊的随机数发生器方法，如同第 7 章的附录 7A 中的图 7.5 和图 7.6 表示的那样。大多数编译器都具备固定的随机数产生器，但其中很多非常简陋以至于不能使用；对于这个问题第 7 章将会全面讨论。

如同在第 1.3.2 小节一般性讨论的那样，方便的做法是将程序模块化为多个子程序，使得逻辑和相互影响变得清晰(即使计算不是最有效)。除了主程序，仿真程序包括初始化、定时、报告生成，以及生成指数随机变数的例程，如图 1.3 所示。如果我们写一个单独的例程去更新连续时间统计，进行$Q(t)$和$B(t)$曲线下的面积累加，它还会使事情简化。然而，最重要的行为发生在事件例程中，我们将事件编号如表 1.1 所示。

表　1.1

事件描述	事件类型
顾客到达系统	1
顾客完成服务后从系统离去	2

图 1.8 所示的包括了到达事件的流程图。首先，生成未来下一个到达的时间并放入事件表中。然后检查服务台以确定是否忙。如果忙，队列中的顾客数加 1，并检测分配给队列的存储空间是否已满(见第 1.4.4 小节的代码)。如果队列满了，则产生一个错误信息并且仿真停止；如果队列中仍有空间，则将新顾客的到达时间置于队列的(新)末尾(如果编程语言支持动态内存分配并使用动态内存分配，队列满的检查可以省略)。另一方面，如果到达的顾客发现服务台闲，则顾客的延时为 0，但仍算一个延时，已完成的顾客延误数加 1，服务台必须置于忙状态，在事件表中安排到达顾客从服务离去的时间。

离去事件的逻辑用图 1.9 所示的流程图描述。回顾一下当一个服务完成(且随后离去)发生时，该子例程被调用。如果顾客离去后队列中无其他顾客，服务台置成闲并且将离去事件排除在外，因为下一个事件必须是到达事件。另一方面，如果顾客离去后尚有一个或多个顾客，那么队列中的第一个顾客将离开队列并进入服务，从而队列长度减 1，并且计算该顾客在队列中的延误时间并写入相应的统计计数器中。延误数加 1，安排当前进入服务的顾客的离去事件。最后，队列的剩余部分(如果有的话)会前移一个位置。本章中，我们的队列表的实现将是非常简单的，但肯定不是最有效的；第 2 章讨论更好的表处理方法，以便对于像队列这样的事情进行建模。

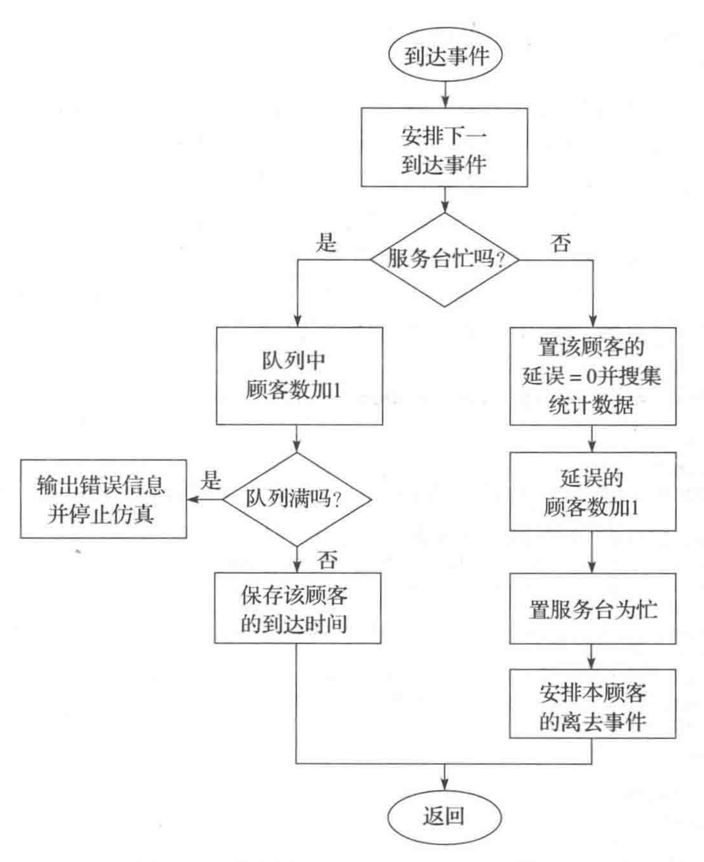


图 1.8 到达例程的流程图——排队模型

在下一节我们会给出一个例子，说明如何用上述流程图来写C语言程序。其结果在第1.4.5小节讨论。这个程序既不是最简单的，也不是最有效的，而代之以设计，是为了说明对更复杂的仿真，人们应该如何组织程序。

1.4.4 C程序

本节给出了 $M/M/1$ 队列仿真的C程序。我们使用C语言的ANSI标准版，这是由Kernigan和Ritchie定义的，特别是使用函数原型。我们还利用C语言的优势，给变量和函数起相当长的名字，因此他们应该是自解释的(例如，仿真时间的当前值是一个变量，称为sim_time)。我们已经在几种不同的计算机和编译器上运行了我们的C程序。由于浮点操作的不精确性，数字结果在某些情况下不相同。这可能很重要，例如，在仿真的某些点，如果两个事件的时间安排非常靠近，舍入误差将导致事件发生顺序不同。必须建立C语言的数学库的连接，这需要根据编译器设置选项。

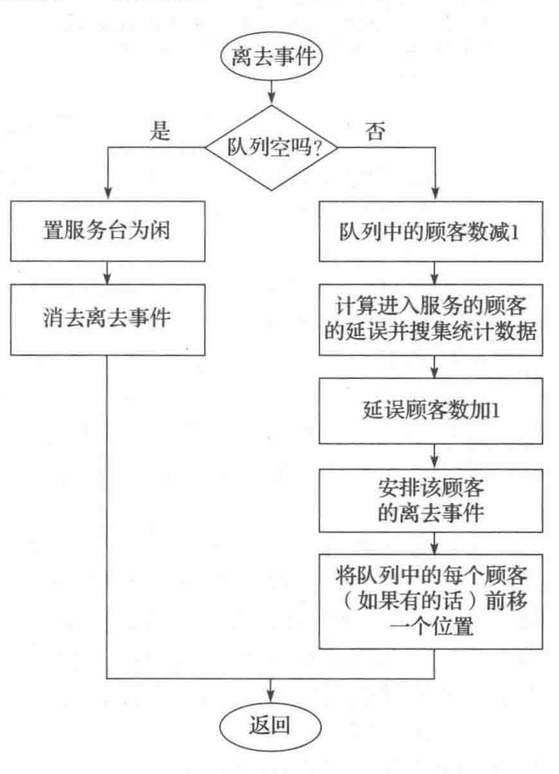


图 1.9 离去例程的流程图——排队模型

全部代码在 www. mhhe. com/law 提供下载。

外部定义在图 1.10 中给出。包含了头文件 lcgrand. h(在图 7.6 中列出)，以声明随机数生成器的函数。符号常数 Q_LIMIT 设为 100，我们猜测(通过实验和错误，它也许必须调整)用作最大队列长度(如同之前提到的，如果我们使用动态分配内存，则能省去这个猜测；当 C 支持动态分配内存时，在我们的例子中就不使用它了)。为了代码的可读性，server_status 变量的两个状态被定义为符号常数 BUSY 和 IDLE。定义文件指针 * infile 和 * outfile 使得我们从代码内部而不是在操作系统级别上打开输入或输出文件。如同我们讨论过的那样，还要注意事件表，它将放在一个叫做 time_next_event 的数组里来实现，数组的第 0 项会被忽略，以确保索引与事件类型相一致。

```
/* External definitions for single-server queueing system. */

#include <stdio.h>
#include <math.h>
#include "lcgrand.h"  /* Header file for random-number generator. */

#define Q_LIMIT 100  /* Limit on queue length. */
#define BUSY      1  /* Mnemonics for server's being busy */
#define IDLE      0  /* and idle. */

int   next_event_type, num_custs_delayed, num_delays_required, num_events,
      num_in_q, server_status;
float area_num_in_q, area_server_status, mean_interarrival, mean_service,
      sim_time, time_arrival[Q_LIMIT + 1], time_last_event, time_next_event[3],
      total_of_delays;
FILE  *infile, *outfile;

void  initialize(void);
void  timing(void);
void  arrive(void);
void  depart(void);
void  report(void);
void  update_time_avg_stats(void);
float expon(float mean);
```

图 1.10 外部定义的 C 代码——排队模型

主函数的代码如图 1.11 所示。输入和输出文件被打开，仿真的事件类型数在本模型中被初始化为 2。输入参数接着从文件 mm1. in 读入，文件中有一行数字：1.0，0.5 和 1 000，中间用空格分隔。在撰写完报告头并回应了输入参数(作为检查它们是否被正确的读取的手段)后，就调用初始化函数。接下来的“while”循环会在满足 1 000 个顾客延误的停止条件之前，使仿真持续运行。在“while”循环内部，定时函数首先被调用，以确定要发生的下一事件的类型并将仿真时钟推进到它的发生时间。在处理此事件之前，更新 $Q(t)$和 $B(t)$曲线下面积的函数会被调用；此时这样做，在处理每一个事件前，我们对这些面积自动更新。然后是开关语句，基于 next_event_type(到达是 1，离去是 2)，将控制转移到相应的事件函数。在“while”循环结束时，报告函数被调用，输入、输出文件关闭，仿真结束。

初始化函数的代码在图 1.12 给出。这里，每一条语句对应着一个图 1.7a 所示计算机表达的一个成分。注意，第一个到达的时间 time_next_event[1]的确定办法是，将一个均值为 mean_interarrival 的指数型随机变量，即 expon(mean_interarrival)加到仿真时间 sim_time=0 上(虽然它的值为 0，但我们还是在这个语句中明确地使用了“sim_time”，以说明确定将来事件的时间的语句的一般形式)。由于在 sim_time=0 时，没有顾客出现，则下一离去时间，即 tim_next_event[2]，被设置为 1.0e+30(C 语言对 10^{30}的表示方法)，保证了第一个事件一定会是到达。

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("mm1.in",  "r");
    outfile = fopen("mm1.out", "w");

    /* Specify the number of events for the timing function. */

    num_events = 2;

    /* Read input parameters. */

    fscanf(infile, "%f %f %d", &mean_interarrival, &mean_service,
           &num_delays_required);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Single-server queueing system\n\n");
    fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
            mean_interarrival);
    fprintf(outfile, "Mean service time%16.3f minutes\n\n", mean_service);
    fprintf(outfile, "Number of customers%14d\n\n", num_delays_required);

    /* Initialize the simulation. */

    initialize();

    /* Run the simulation while more delays are still needed. */

    while (num_custs_delayed < num_delays_required) {

        /* Determine the next event. */

        timing();

        /* Update time-average statistical accumulators. */

        update_time_avg_stats();

        /* Invoke the appropriate event function. */

        switch (next_event_type) {
            case 1:
                arrive();
                break;
            case 2:
                depart();
                break;
        }
    }

    /* Invoke the report generator and end the simulation. */

    report();

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

图 1.11　主函数的 C 代码——排队模型

定时函数在图 1.13 中给出，它用于比较 time_next_event[1]，time_next_event[2]，… time_next_event[num_events]（回忆一下，num_events 是在主函数中设置的），并设置

next_event_type 等于发生时间最小的事件类型。万一有结，则选择最小编号事件类型。然后仿真时钟推进到所选事件类型 mean_time_next_event 的发生时间。该程序稍微有点复杂，因为有对事件表是否为空的错误检查；我们定义的事件表为空的含义是所有事件安排在 time$=10^{30}$发生，如果有此情况出现(如 next_event_type=0 所指示的)，则产生带有当前仿真钟时间的错误信息(作为一个可能的调试帮助手段)，仿真终止。

```
void initialize(void)  /* Initialization function. */
{
    /* Initialize the simulation clock. */

    sim_time = 0.0;

    /* Initialize the state variables. */

    server_status   = IDLE;
    num_in_q        = 0;
    time_last_event = 0.0;

    /* Initialize the statistical counters. */

    num_custs_delayed  = 0;
    total_of_delays    = 0.0;
    area_num_in_q      = 0.0;
    area_server_status = 0.0;

    /* Initialize event list.  Since no customers are present, the departure
       (service completion) event is eliminated from consideration. */

    time_next_event[1] = sim_time + expon(mean_interarrival);
    time_next_event[2] = 1.0e+30;
}
```

图 1.12　初始化函数的 C 代码——排队模型

```
void timing(void)  /* Timing function. */
{
    int   i;
    float min_time_next_event = 1.0e+29;

    next_event_type = 0;

    /* Determine the event type of the next event to occur. */

    for (i = 1; i <= num_events; ++i)
        if (time_next_event[i] < min_time_next_event) {
            min_time_next_event = time_next_event[i];
            next_event_type     = i;
        }

    /* Check to see whether the event list is empty. */

    if (next_event_type == 0) {

        /* The event list is empty, so stop the simulation. */

        fprintf(outfile, "\nEvent list empty at time %f", sim_time);
        exit(1);
    }

    /* The event list is not empty, so advance the simulation clock. */

    sim_time = min_time_next_event;
}
```

图 1.13　定时函数的 C 代码——排队模型

到达事件函数的代码在图1.14给出，接下来的讨论在第1.4.3小节和流程图1.8给出。注意，“sim_time”是刚到的顾客的到达时间，而且通过查看num_in_q当前是否大于Q_LIMIT进行队列溢出检查，数组time_arrival的维数就是其长度。

```
void arrive(void)  /* Arrival event function. */
{
    float delay;

    /* Schedule next arrival. */

    time_next_event[1] = sim_time + expon(mean_interarrival);

    /* Check to see whether server is busy. */

    if (server_status == BUSY) {

        /* Server is busy, so increment number of customers in queue. */

        ++num_in_q;

        /* Check to see whether an overflow condition exists. */

        if (num_in_q > Q_LIMIT) {

            /* The queue has overflowed, so stop the simulation. */

            fprintf(outfile, "\nOverflow of the array time_arrival at");
            fprintf(outfile, " time %f", sim_time);
            exit(2);
        }

        /* There is still room in the queue, so store the time of arrival of the
           arriving customer at the (new) end of time_arrival. */

        time_arrival[num_in_q] = sim_time;
    }

    else {

        /* Server is idle, so arriving customer has a delay of zero.  (The
           following two statements are for program clarity and do not affect
           the results of the simulation.) */

        delay            = 0.0;
        total_of_delays += delay;

        /* Increment the number of customers delayed, and make server busy. */

        ++num_custs_delayed;
        server_status = BUSY;

        /* Schedule a departure (service completion). */

        time_next_event[2] = sim_time + expon(mean_service);
    }
}
```

图1.14 到达函数的C代码——排队模型

离去事件函数的代码如图1.15所示，它是当一个服务行为完成(接着离去)发生时由主程序调用的，其逻辑与流程图(见图1.9)在第1.4.3小节中讨论过。注意，如果在“else”前省略语句“time_next_event[2]=1.0e+30;”，则程序将进入死循环(为什么?)。靠近函数末尾的“for”循环将其余的队列(如果有的话)前移一位，保证进入服务(在队列中被延误后)的下一个顾客的到达时间总是存入time_arrival[1]。注意，如果当前队列为

空(例如，刚刚离开队列进入服务的顾客是队列中的唯一顾客)，则 num_in_q=0，此循环完全不会执行，因为循环变量 i(开始值为 1)的初始值已经超过其终值(num_in_q=0)(用这种简单方法来管理队列肯定是低效的，但可以通过使用指针来改进；我们在第 2 章会回到这个问题)。对于离去的最后注释包括：从仿真钟的值 sim_time 减去 time_arrival[1]得到队列中的延误时间。如果仿真运行的时间很长，那么 sim_time 和 time_arrival[1]与它们之间的差相比，是非常大的数；这样，由于他们都被存储为有限精度的浮点数，当做这样的减法时，有可能会发生精度严重丢失问题。因为这个原因，当我们要运行仿真很长时间的时候，对 sim_time 和数组 time_arrival 使用 double 类型是必要的。

```
void depart(void)  /* Departure event function. */
{
    int   i;
    float delay;

    /* Check to see whether the queue is empty. */

    if (num_in_q == 0) {

        /* The queue is empty so make the server idle and eliminate the
           departure (service completion) event from consideration. */

        server_status      = IDLE;
        time_next_event[2] = 1.0e+30;
    }

    else {

        /* The queue is nonempty, so decrement the number of customers in
           queue. */

        --num_in_q;

        /* Compute the delay of the customer who is beginning service and update
           the total delay accumulator. */

        delay            = sim_time - time_arrival[1];
        total_of_delays += delay;

        /* Increment the number of customers delayed, and schedule departure. */

        ++num_custs_delayed;
        time_next_event[2] = sim_time + expon(mean_service);

        /* Move each customer in queue (if any) up one place. */

        for (i = 1; i <= num_in_q; ++i)
            time_arrival[i] = time_arrival[i + 1];
    }
}
```

图 1.15　离去函数的 C 代码——排队模型

图 1.16 给出报告函数的代码，当主程序中的“while”循环结束时调用此代码。用观察到延误的顾客数去除总延误时间，得到平均延时；队列中的时间平均数用 $Q(t)$下的面积除以终止时的仿真钟值得到，$Q(t)$下的面积现已更新至仿真的结束(由于更新面积的函数是由主程序调用的，在处理到达或离开之前，其中一个会结束仿真)。服务台利用率的计算是用 $B(t)$下的面积除以最终仿真钟时间得到，所有三个度量值均被直接写出。我们也将最终仿真钟本身的值写出，看看需要多久才能观察到 1 000 个延误。

图 1.17 表示了函数 update_time_avg_stats。此函数的调用是在处理每一个事件(任意类型)之前，它更新两函数下的面积，以满足连续时间统计的需要。该例程只是为了编写代码的方便而分离出来，而不是一个事件例程。首先计算自上次事件以来的时间，然后再将上次

事件的时间代入到当前的时间，以便为下一次进入此函数做好准备。队列中顾客数量函数下的面积要增加，增加量为从上一个事件以来时间区间上 $Q(t)$下的长方形面积，区间的宽度为 time_since_last_event，高为 mun_in_q；记住，此函数应在处理事件之前调用，并且像 num_in_q这样的状态变量仍然保留以前的值。然后 $B(t)$下的面积要增加，增加量是宽为 time_since_last_event，高为 server_status 的长方形的面积；这就是为什么将 server_status 定义为 0 或 1 很方便的原因。注意，该函数，包括了两个浮点数的减法(sim_time－time_last_event)，例如离去，有可能因为仿真运行时间过长而导致其差与其本身差距过大；在此种情况下，需要将 sim_time 与 time_last_event 两个变量声明为 double 类型。

```
void report(void)  /* Report generator function. */
{
    /* Compute and write estimates of desired measures of performance. */

    fprintf(outfile, "\n\nAverage delay in queue%11.3f minutes\n\n",
            total_of_delays / num_custs_delayed);
    fprintf(outfile, "Average number in queue%10.3f\n\n",
            area_num_in_q / sim_time);
    fprintf(outfile, "Server utilization%15.3f\n\n",
            area_server_status / sim_time);
    fprintf(outfile, "Time simulation ended%12.3f minutes", sim_time);
}
```

图 1.16 报告函数的 C 代码——排队模型

```
void update_time_avg_stats(void)  /* Update area accumulators for time-average
                                     statistics. */
{
    float time_since_last_event;

    /* Compute time since last event, and update last-event-time marker. */

    time_since_last_event = sim_time - time_last_event;
    time_last_event       = sim_time;

    /* Update area under number-in-queue function. */

    area_num_in_q      += num_in_q * time_since_last_event;

    /* Update area under server-busy indicator function. */

    area_server_status += server_status * time_since_last_event;
}
```

图 1.17 update_time_avg_stats 函数的 C 代码——排队模型

函数 expon 在图 1.18 给出，它可以生成均值 β= mean(传到 expon)的指数随机变量，接下来的是在第 1.4.3 小节中讨论过的算法。随机数发生器 lcgrand，此处用到了整型自变量 1，将在第 7 章中详细讨论，并在图 7.5 中专门给出。C 语言预定义函数 log 返回其自变量的自然对数。

```
float expon(float mean)  /* Exponential variate generation function. */
{
    /* Return an exponential random variate with mean "mean". */

    return -mean * log(lcgrand(1));
}
```

图 1.18 expon 函数的 C 代码——排队模型

此处讨论的程序必须结合图 7.5 给出的随机数发生器代码才能运行。要做到这些，需要进行分别编译，采用与安装依赖的方式将对象代码连接在一起。

1.4.5 仿真输出与讨论

图 1.19 给出了输出结果(文件名为 mm1.out)。在这次运行中，队列中平均延误时间为 0.430 分钟，队列中平均有 0.418 个顾客，服务台 46%的时间忙。花了1 027.915 仿真分钟(simulated minutes)使得仿真运行至完成 1 000 个延误，这个结果看起来是合理的，因为两个顾客到达之间的期望时间为 1 分钟(平均延误，队列中平均人数，以及利用率在此模型中都如此接近，这种情况不总是这样，参见附录 1B)。

```
Single-server queueing system

Mean interarrival time          1.000 minutes

Mean service time               0.500 minutes

Number of customers              1000



Average delay in queue          0.430 minutes

Average number in queue         0.418

Server utilization              0.460

Time simulation ended        1027.915 minutes
```

图 1.19 输出报告——排队模型

注意，输出中这些特定的数据，在根源上，是由本次随机数发生器所决定的。如果使用一个不同的随机数发生器，或者我们用另外一种方式使用此随机数发生器(例如用其他“种子”或“流”，如同在第 7 章讨论的那样)，那么输出中会产生不同的数据。因此，这些数据不能当做“答案”，而只能是我们想知道的那些量的“估计”，例如 $d(n)$，$q(n)$和$u(n)$(而且有可能是坏的估计)；仿真输出数据的统计分析在第 9 章到第 12 章讨论。同样，结果也是输入参数的函数，在此情况下，平均到达时间间隔，服务时间和 $n=1\,000$ 的终止条件，均会受我们进行仿真初始化的方法(空和闲)的影响。

在某些仿真研究中，我们可能要估计模型的稳态特性(参见第 9 章)，即仿真运行很长时间(理论上为无穷)后模型的特征。对于我们已经正在讨论的简单 $M/M/1$ 队列来说，能解析地计算出队列中稳态平均延误时间、队列中稳态时间平均顾客数，以及服务台稳态利用率，所有这些性能度量值都为 0.5[例如，参见 Ross(2003，第 480-487 页)]。这样，如果我们想确定这些稳态度量值，我们基于 $n=1\,000$ 个延误这种停止策略的估计，绝对地说，至少不会偏差太远。然而，我们是有点儿运气，因为 $n=1\,000$ 是任意选择的！实际上，为给出稳态度量的一个好的估计，停止规则的选择是相当困难的。为了说明这一点，假设 $M/M/1$ 队列的顾客到达速率从 1 个/分钟增加为 1.98 个/分钟(现在的平均到达间隔时间为 0.505 分钟)，平均服务时间不变，如同前面一样，我们希望在运行长度 $n=1\,000$ 个延误条件下估计稳态度量值。我们执行这个仿真运行，得到平均延误时间、队列中平均人数、服务台利用率分别为 17.404 分钟、34.831 人、0.997。由于这些度量的真实稳态值分别是 49.5 分钟、98.01 人、0.99，很明显，停止规则不能任意选择。我们将在第 9 章讨论如何对稳态仿真规定运行长度。

读者也许会感到奇怪，为什么我们不估计顾客在系统中的期望平均等待时间 $w(n)$，而是队列中的期望平均延误时间 $d(n)$，这里顾客等待时间定义为顾客从到达的时刻至顾客服务完成并离去的时刻这一时段。有两个原因，第一，对很多排队系统来说，我们认为，顾客等待其他顾客接受服务而在队列中延误是顾客在系统中等待的最烦人的部分。此外，如果队列代表制造系统的一部分，这里“顾客”是在机器(“服务台”)处等待服务的零件，于是队列中延误时间表示一种损失，而服务中花费的时间是“必需的”。我们关注队列中延误时间的第二个原因是统计效率。$w(n)$的通常估计是：

$$\hat{w}(n)=\frac{\sum_{i=1}^{n}W_i}{n}=\frac{\sum_{i=1}^{n}D_i}{n}+\frac{\sum_{i=1}^{n}S_i}{n}=\hat{d}(n)+\overline{S}(n) \tag{1.7}$$

其中，$W_i=D_i+S_i$是第 i 个顾客在系统中的等待时间；$\overline{S}(n)$是 n 个顾客服务时间的平均值。

首先，由于为了方便执行仿真，服务时间的分布必须已知，那么期望的或平均服务时间 $E(S)$也是已知的，$w(n)$的另一个估计是：

$$\widetilde{w}(n) = \hat{d}(n) + E(S)$$

注意，式(1.7)中 $\overline{S}(n)$是 $E(S)$的一个无偏估计。在几乎在所有的排队仿真中，较之$\hat{w}(n)$，$\widetilde{w}(n)$是 $w(n)$的一个更有效(变化更小)的估计，所以它更好(两个估计量都是无偏的)。这样，如果人们想要 $w(n)$的估计，则估计 $d(n)$，再加上已知的期望服务时间$E(S)$。一般来说，一旦可能，确实就用它们的期望值去替代估计值(见第 11.5 节间接估计的讨论)。

1.4.6　其他终止规则

在上述的排队例子中，仿真在延误的顾客为 1 000 时停止；那么最后的仿真钟的值是一个随机变量。然而，对于很多实际的模型来说，仿真在某一个固定的时间停止，比如说 8 小时。由于我们例子中的到达间隔和服务时间是连续随机变量，故仿真恰好在 480 分钟后停止的概率为 0(忽略计算机的精度有限性)。因此，为了在一个规定的时间停止仿真，我们引入一个虚构的“结束仿真”事件(称为类型 3 事件)，安排它在 480 分钟时发生。当此事件发生的时间(保存在事件表的第 3 个位置上)比事件列表中所有其他项都早时，就调用报告生成器，从而仿真终止。延误的顾客数现在是一个随机变量。

在程序内可以实现这些想法，办法是改变外部定义，主函数，以及初始化和报告函数，如图 1.20～图 1.23 所示；程序的其他的部分不需要改动。在图 1.20 和图 1.21 中，注意，我们现在有 3 个事件，即要求的仿真运行长度 time_end，现在是一个输入的参数(num_delays_required 已经去掉)；而“switch”语句也改了。为了停止仿真，原来的“while”循环已经被图 1.21 所示程序的“do while”循环替代，这里，循环本身不断重复，直到刚刚执行的事件类型是 3 为止(仿真停止)；在选择了执行事件类型 3 后，循环停止且仿真结束。在主程序(如前)中，我们在进入事件函数前调用了 update_time_avg_stats，从而，在类型 3 事件(结束仿真)为下一个事件时，尤其要注意，在仿真结束时要修改面积。图 1.22 所示程序的初始化函数的唯一改变就是加入了一个语句 time_next_event[3]3＝time_end，它安排仿真的终止。图 1.23 中的报告函数的唯一改变是写出延时的顾客总数而不是仿真结束的时间，因为在这种情况下，我们知道终止时间是 480 分钟，但不知道在该时间内有多少顾客延误完成了。

```c
/* External definitions for single-server queueing system, fixed run length. */

#include <stdio.h>
#include <math.h>
#include "lcgrand.h"  /* Header file for random-number generator. */

#define Q_LIMIT 100  /* Limit on queue length. */
#define BUSY      1  /* Mnemonics for server's being busy */
#define IDLE      0  /* and idle. */

int   next_event_type, num_custs_delayed, num_events, num_in_q, server_status;
float area_num_in_q, area_server_status, mean_interarrival, mean_service,
      sim_time, time_arrival[Q_LIMIT + 1], time_end, time_last_event,
      time_next_event[4], total_of_delays;
FILE  *infile, *outfile;

void  initialize(void);
void  timing(void);
void  arrive(void);
void  depart(void);
void  report(void);
void  update_time_avg_stats(void);
float expon(float mean);
```

图　1.20

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("mm1alt.in",  "r");
    outfile = fopen("mm1alt.out", "w");

    /* Specify the number of events for the timing function. */

    num_events = 3;

    /* Read input parameters. */

    fscanf(infile, "%f %f %f", &mean_interarrival, &mean_service, &time_end);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Single-server queueing system with fixed run");
    fprintf(outfile, " length\n\n");
    fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
            mean_interarrival);
    fprintf(outfile, "Mean service time%16.3f minutes\n\n", mean_service);
    fprintf(outfile, "Length of the simulation%9.3f minutes\n\n", time_end);

    /* Initialize the simulation. */

    initialize();

    /* Run the simulation until it terminates after an end-simulation event
       (type 3) occurs. */

    do {

        /* Determine the next event. */

        timing();

        /* Update time-average statistical accumulators. */

        update_time_avg_stats();

        /* Invoke the appropriate event function. */

        switch (next_event_type) {
            case 1:
                arrive();
                break;
            case 2:
                depart();
                break;
            case 3:
                report();
                break;
        }

    /* If the event just executed was not the end-simulation event (type 3),
       continue simulating.  Otherwise, end the simulation. */

    } while (next_event_type != 3);

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

图 1.21

```
void initialize(void)  /* Initialization function. */
{
    /* Initialize the simulation clock. */

    sim_time = 0.0;

    /* Initialize the state variables. */

    server_status   = IDLE;
    num_in_q        = 0;
    time_last_event = 0.0;

    /* Initialize the statistical counters. */

    num_custs_delayed  = 0;
    total_of_delays    = 0.0;
    area_num_in_q      = 0.0;
    area_server_status = 0.0;

    /* Initialize event list.  Since no customers are present, the departure
       (service completion) event is eliminated from consideration.  The end-
       simulation event (type 3) is scheduled for time time_end. */

    time_next_event[1] = sim_time + expon(mean_interarrival);
    time_next_event[2] = 1.0e+30;
    time_next_event[3] = time_end;
}
```

图 1.22

```
void report(void)  /* Report generator function. */
{
    /* Compute and write estimates of desired measures of performance. */

    fprintf(outfile, "\n\nAverage delay in queue%11.3f minutes\n\n",
            total_of_delays / num_custs_delayed);
    fprintf(outfile, "Average number in queue%10.3f\n\n",
            area_num_in_q / sim_time);
    fprintf(outfile, "Server utilization%15.3f\n\n",
            area_server_status / sim_time);
    fprintf(outfile, "Number of delays completed%7d",
            num_custs_delayed);
}
```

图 1.23

输出文件(名字为 mmlalt. out)在图 1.24 中给出。在本次运行中，完成的顾客延误数为 475，这似乎是合理的，因为顾客以每分钟 1 个的平均速率到达，仿真运行 480 分钟。性能的三个同样的度量又一次在数值上互相接近，但前两个较 1 000 个延误的仿真中的值略微低一些。对此，一个有可能的原因是目前的运行长度大约只有之前的一半那么长，并且由于仿真初始条件是空且闲(非堵塞状态)，在较短的运行中模型出现堵塞的机会少一些。然而，我们再次强调，这只是一次运行，且受到很多个不确定性的影响；仅从一次的运行来评价这种不确定性的程度是不容易的。

如果所考虑的排队系统实际上是一个只有一个理发师的从早 9 点到晚 5 点开门的理发店，在准确的 8 小时后停止仿真，这则意味着可能会有某位顾客的头发只理了一半。在这种情况下，我们会希望在 8 小时后理发店关门，但继续进行仿真，直到关门时所有在店的顾客(如果有的话)服务完为止。在习题 1.10 中，要求读者提供为实施此停止规则所必需的程序改变(也可参见第 2.6 节)。

```
Single-server queueing system with fixed run length

Mean interarrival time        1.000 minutes

Mean service time             0.500 minutes

Length of the simulation    480.000 minutes


Average delay in queue        0.399 minutes

Average number in queue       0.394

Server utilization            0.464

Number of delays completed      475
```

图 1.24

1.4.7 事件和变量的确定

在第 1.3 节中我们将事件定义为可能改变系统状态的瞬时发生的行为，而在第 1.4.1 小节的简单的单服务台排队中，识别事件并不困难。然而，特别是对于复杂系统，有时会出现对一个模型如何一般性地确定事件的个数和定义的问题。为保证仿真按正确的事件顺序运行并且得到想要的输出量度，定义所需要的状态变量也是困难的。没有一个完全通用的方法来回答这些问题，不同的人也许会用不同的办法根据事件和变量来表示一个模型，而且都可能正确。但总有一些规律和技巧，以帮助规范模型结构并且避免逻辑错误。

Schruben(1983b)给出了事件图方法，后来 Sargent(1988)以及 Som 和 Sargent(1989)对其进行了细化和扩展。在此方法中，建议的事件，每个用节点来表示，用有向弧(箭头)连接起来，用于表示事件是如何是由其他事件或自己来确定时间的。举例来说，在第 1.4.3 小节的排队仿真中，到达事件确定了其自身和(可能)离去事件的未来发生的时间，离去事件也许会确定其自身另一个未来发生的时间；另外，初始化必须安排到达事件，使得仿真可以开始。事件图将建议的事件(节点)集合用弧连接起来，弧指示出那些能确定发生时间的事件的类型。图 1.25 给出了我们单服务台排队系统的事件图，其中粗而光滑线的箭头表示箭头末端的事件可以由箭头起端的事件来确定一个数量(可能)非零的时间。细的波浪线箭头表示在其末端的事件的时间首先确定。因此，到达事件自己重调度，并可以调度离去事件(在一个到达发现服务台闲的情况下)，而离去事件也可自己重调度(如果一个顾客离去后队列中还有其他人)。

对于此模型，有人也许会问为什么我们没有清晰地把顾客进入服务(或者从队列里或是从到达)的行为当作一个独立的事件。这当然可以，而且可以导致状态的改变(也就是，队伍的长度减 1)。实际上，这可以当做一个独立的事件，而不会导致仿真的错误，图 1.26 给出了这种情况的事件图。两个细而平滑线的箭头每个表示箭头始端的事件，隐含着安排箭头终端的事件的时间，中间没有时间插入，即立即；在这种情况下，细直的平滑线箭头表示一个到达空系统的顾客，该顾客“进入服务”事件会被立即调度发生，细曲的平滑线箭头表示身后仍有队列等待的一个顾客离去，那么队列中第一个顾客将立即调度进入服务。事件的个数现在已经增加了 1，则我们的模型表示就稍微复

图 1.25 事件图，排队模型

图 1.26 事件图与具有独立的“进入服务”事件的排队模型

杂一些。事件图的一个用途是通过消除不必要的事件来简化仿真事件结构。有几个用于简化的“规则”，其中一个就是：如果一个事件节点有入弧，这些入弧全部是细平的(也就是，该事件被调度的唯一方法是由其他事件引起且无中间插入的时间)，那么这个事件就可以从模型中去除，它的行为将合并到以 0 时间调度它的事件中。在这里，“进入服务”事件就可以去除，其行为一部分放到到达事件(当顾客到达一个闲服务台并马上开始服务时)，一部分放到离去事件(当顾客完成服务且有一个队列，下一个顾客从该队列出来进入服务时)；这就将我们带回到图 1.25 所示的较简单的事件图。基本上，只有与其他事件关联的“事件”不需要出现在模型中。减少事件的数量不仅仅简化了模型的概念化，而且也许还加速其执行。然而，必须小心的是以这种方法来“拆分”事件时，处理优先级和时间约束要适当。

另一个简化的规则是有关初始化的。将事件图分解为强连接组件，在每个组件内部，按照弧指示的方向，可以从任意一个节点走遍所有的节点。图 1.25 所示的事件图分解为两个强连接组件(一个组件一个节点)，而图 1.26 所示的事件图有两个强连接组件(其中一个是到达节点本身，另外一个由进入服务节点和离去节点组成)。初始化规则说，在节点的任意强连接组件中，如果没有来自组件外的其他事件节点的入弧，则至少有一个初始化调度的节点；如果违反了这个规则，将不可能执行该组件中的任何事件。在图 1.25 和图 1.26 中，到达节点是这样的强连接组件，因为它没有来自其他节点的入弧，因此必须初始化。图 1.27给出了第 1.4.6 小节中带有固定运行长度的排队模型的事件图，为此我们引入过虚构的“结束仿真”事件。注意这个事件本身是一个没有任何入弧的强连接组件，因此它必须初始化，即仿真结束作为初始化的一部分加以调度。若不然将导致仿真的错误终止。

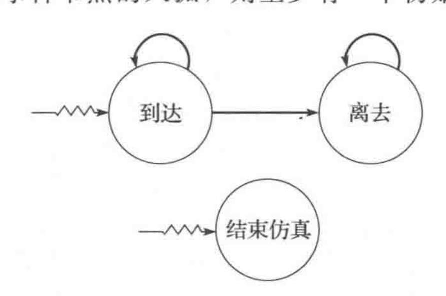


图 1.27　事件图与固定运行长度的排队模型

我们只提供了一部分且是非常简单的事件图技术。还有其他一些特性，包括事件取消关系、相似事件合并为一的方法、细化事件调度弧使之包括条件调度，以及合并需要的状态变量等，这可参见 Schruben(1983b)原文。Sargent(1988)以及 Som 和 Sargent(1989)扩展并细化了该技术，给出了包括柔性制造系统与计算机网络模型的全面阐述。事件图也可用于检测两个看似不同的模型是否可能实际上是等价的(Yücesan 和 Schruben，1992)，以及预测一个模型在执行时是如何增强计算的(Yücesan 和 Schruben，1998)。Schruben 和 Schruben(2004)提供了一个软件包 SIGMA，可以在屏幕上建立表示仿真模型的事件图，然后生成代码并运行模型。Buss(1996)给出了通用事件图综述及教程，Schruben 等人(2003)介绍了事件图的高级应用。

在系统建模时，事件图技术可用于简化结构，以及探测某些类型的错误，在含有大量相互关联事件的复杂模型中特别有用。还应该记住其他一些需要考虑的事，例如，不断地询问为什么需要一个特殊的状态变量，参见习题 1.4。

1.5　库存系统的仿真

现在我们将看看仿真如何用于比较库存系统的不同订货策略。我们模型中的很多元素的功能是实际库存系统中的元素的描述。

1.5.1　问题描述

只销售单一产品的公司希望决定后 n 个月(n 是固定的输入参数)中每一个月的库存应该有多少件。需求的间隔时间是均值为 0.1 个月的独立同分布的指数随机变量。需求量 D 是独立同分布的随机变量(与需求发生时间是独立的)，其中，

$$D=\begin{cases}1, & \text{w. p. } \dfrac{1}{6}\\ 2, & \text{w. p. } \dfrac{1}{3}\\ 3, & \text{w. p. } \dfrac{1}{3}\\ 4, & \text{w. p. } \dfrac{1}{6}\end{cases}$$

其中，w. p. 表示“概率为”(with probability)。

在每个月的开始，公司评测库存水平并决定到底从其供应商订购多少件。如果公司订购了 Z 件，则花费为 $K+iZ$，其中，$K=\$32$，为准备成本，而 $i=\$3$ 为每件订货的边际成本(如果 $Z=0$，则无任何花费)。当下订单后，订货到达所需时间(称为交货延误或提前期)是一个在 0.5 至 1 个月之间均匀分布的随机变量。

公司使用了固定的(s, S)策略以决定订购多少，即

$$Z=\begin{cases}S-I, & I<s\\ 0, & I\geqslant s\end{cases}$$

其中，I 是月初的库存水平。

当需求发生时，如果库存水平至少与需求量一样，则此需求会立即满足。如果需求超过库存水平，需求超过供应的部分需要订货，并在以后的交付中满足(在这种情况下，新的库存水平等于旧的库存水平减去需求量，导致库存水平为负值)。当订货到达时，订货首先用来尽可能的冲销未结的数目(如果有的话)；剩下的订货(如果有的话)才会加到库存。

到目前为止，我们只讨论了一类库存系统的成本，即订货成本。然而，大多数实际库存系统还有两类附加成本，储备成本和短缺成本，我们再介绍一些附加的符号后来讨论它们。令 $I(t)$是 t 时刻的库存水平(注意，$I(t)$可能为正、负，或 0)，令 $I^+(t)=\max\{I(t), 0\}$为 t 时刻库存现货件数[注意，$I^+(t)\geqslant 0$]；并令 $I^-(t)=\max\{-I(t), 0\}$为 t 时刻的短缺量[注意，同样 $I^-(t)\geqslant 0$]。一种 $I(t)$，$I^+(t)$和 $I^-(t)$可能的实现在图 1.28 中给出。$I(t)$减少的时间点即需求发生的时间点。

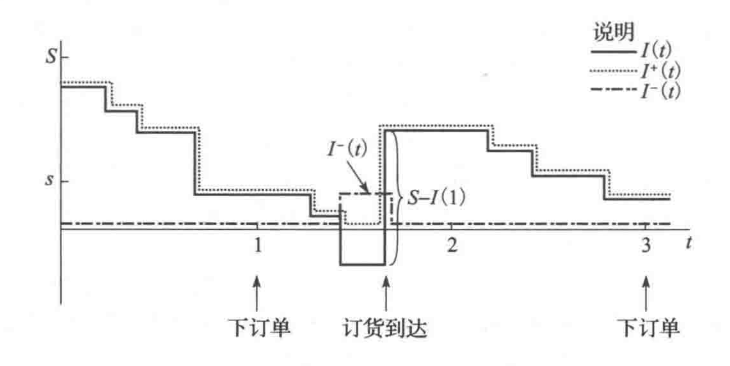


图 1.28 $I(t)$，$I^+(t)$和 $I^-(t)$随时间变化的一个实现

对于我们的模型，假设公司的储备成本是存储在仓库(正)的货物每件每月 $h=\$1$，储备成本包括仓库租金、保险、税和维护费用，以及将资金用于仓库而不是投资到其他地方的机会成本。在我们的公式中，忽略了这样一个事实：即当 $I^+(t)=0$ 时，储备成本仍然存在。然而，由于我们的目标是比较订货策略，忽略这个因素，该因素完全与使用的策略无关，将不会影响我们对哪一个策略更好的评估。这里，由于 $I^+(t)$是在 t 时刻库存的件数，在 n 个月期间库存的时间平均(每月)件数为：

$$\overline{I}^{+}=\frac{\int_{0}^{n} I^{+}(t)\mathrm{d}t}{n}$$

它类似于与第 1.4.1 小节中给出的队列中时间平均顾客数，于是，每月的平均储备成本为 $h\overline{I}^{+}$。

类似地，假设公司的短缺成本是每件每月 $\pi=\$5$；这是考虑到当存在短缺时的额外记录保持费用，以及客户的商誉损失。短缺的时间平均件数为：

$$\overline{I}^{-}=\frac{\int_{0}^{n} I^{-}(t)\mathrm{d}t}{n}$$

这样，每月平均短缺成本是 $\pi\overline{I}^{-}$。

假设初始库存水平为 $I(0)=60$，并且无订货出现。我们对此库存系统做 $n=120$ 个月的仿真，并利用每月平均总成本(是每月平均订货成本、每月平均储备成本和每月平均短缺成本之和)来比较表 1.2 所示 9 个库存策略。

表 1.2

s	20	20	20	20	40	40	40	60	60
S	40	60	80	100	60	80	100	80	100

对于如何考虑选择这些特定的策略的问题我们这里先不做解释，在第 12 章讨论做这种决策的统计技术。

我们注意到这个库存系统仿真模型的状态变量有库存水平 $I(t)$，从公司提交到供应商的未偿订单的数量，以及上一事件的时间[需要该时间计算 $\overline{I}^{+}(t)$和 $\overline{I}^{-}(t)$函数下的面积]。

1.5.2 程序组织和逻辑

我们的库存系统模型使用了表 1.3 所示的事件类型。

表 1.3

事件描述	事件类型
订货从供应商到公司的到达	1
顾客对产品的需求	2
n 个月后仿真结束	3
月初库存检查(以及可能的订货)	4

我们选择让仿真结束事件是类型 3 而不是类型 4，这是因为在 120 时刻，“仿真结束”和“库存检查”两者最终都要执行，而我们希望此时先执行前一个事件(因为当仿真时间超过 120 的时候，库存检查和可能的订货将变得没有意义，订货引起的订货成本再也不会到达)。事件类型 3 先于事件类型 4 执行是有保障的，因为即使有两个或多个事件同时发生，定时例程先执行最小事件类型数的事件。一般说来，一个仿真模型的设计应当考虑发生时间求解约束时要以合适的顺序来处理事件。图 1.29 给出了事件图(见第 1.4.7 小节)。

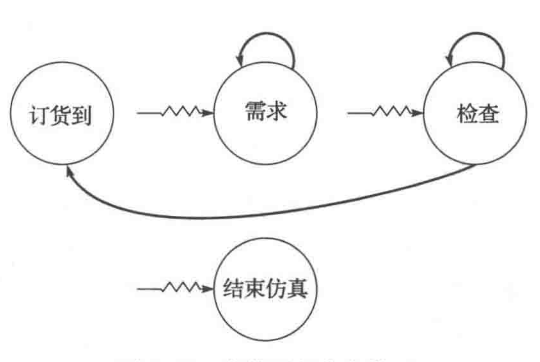


图 1.29 事件图与库存模型

为了仿真该系统，需要三种随机变量。需求间隔时间是指数分布的，从而与第 1.4 节中开发的相同的算法(和代码)也可以在此使用。如上所述，需求量随机变量 D 必须是离散

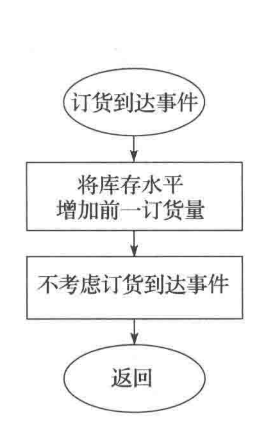
的，而且可用如下的方法生成。首先，把单位区间分为连续的子区间 $C_1=\left[0, \frac{1}{6}\right)$，$C_2=\left[\frac{1}{6}, \frac{1}{2}\right)$，$C_3=\left[\frac{1}{2}, \frac{5}{6}\right)$，$C_4=\left[\frac{5}{6}, 1\right]$，并从随机数发生器得到一个 $U(0, 1)$的随机变量 U。如果 U 落在 C_1 中，返回 $D=1$；如果 U 落在 C_2 中，则返回 $D=2$；依此类推。因为 C_1 的宽度是 $\frac{1}{6}-0=\frac{1}{6}$，还因为 U 是$[0, 1]$上均匀分布的，U 落在 C_1 中(从而我们返回 $D=1$)的概率是 $\frac{1}{6}$；这与希望 $D=1$ 的概率是一致的。类似地，如果 U 落在 C_2 中的概率等于 C_2 的宽度 $\frac{1}{2}-\frac{1}{6}=\frac{1}{3}$，如要求的那样，我们返回 $D=2$；其他的区间依此类推。生成需求量的子程序遵循这个原则，并将定义上述子区间的截点作为输入，这些截点是 D 的分布的累积概率。

交货延时是均匀分布的，但不在单位区间$[0, 1]$上。一般说来，我们可以通过生成一个 $U(0, 1)$的随机数 U，然后返回 $a+U(b-a)$来生成任意区间$[a, b]$上均匀分布的随机变量。这个方法的正确性直观上似乎是显然的，但将在第 8.3.1 小节给出正式的证明。

我们现在描述事件类型 1，2 和 4 的逻辑，它实际上包括状态变化。

图 1.30 给出了订货到达事件的流程图，当订货(之前就下了订单)从供应商到达时，订货到达事件必然造成必要的变化。库存水平增加了订购的数量，而且必须不再考虑订货到达事件(对具有同样参数的本模型，考虑在某一时刻是否可能有一个以上的订货出现的问题，见习题 1.12)。

需求事件的流程图如图 1.31 所示，该流程图处理了必要的变化，以表示需求的发生。首先，产生需求量，库存减少该需求量，最后，下一需求的时间安排到事件表中。注意，这里库存水平可能变为负值。



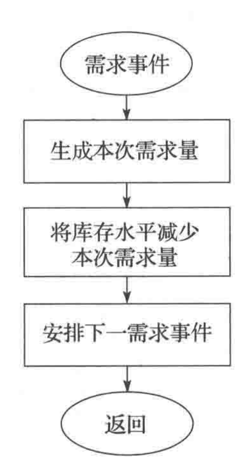




图 1.30 订货到达例程的流程图与库存模型

图 1.31 需求例程的流程图与库存模型

库存检查事件在每个月的月初发生，图 1.32 给出了它的流程图。如果在检查时的库存水平 $I(t)$至少为 s，则无订货发生，并且除了将下一个检查安排到事件表中外，将不做任何事情。另一方面，如果 $I(t)<s$，我们就会希望下一个数量为 $S-I(t)$件的订单。若想做到这一点，方法是把订货数量$[S-I(t)]$保存起来，直到该订货到达为止，并安排好它到达时间。在此情况下，我们要安排下一个库存检查事件。

如同在单服务台排队模型中那样，写一个独立的非事件的程序去更新连续时间统计累加器是方便的。然而对于这个模型，这样做就稍显复杂，所以我们在图 1.33 中给出该活动的流程图。主要的问题是我们是要更新 $I^-(t)$还是 $I^+(t)$下的面积(或均不需要)。如果自上一事

件以来的库存水平成为负的，那么，我们就出现短缺，从而只有 $I^-(t)$下的面积应该更新；另一方面，如果库存水平是正的，我们只需更新 $I^+(t)$下的面积。如果库存水平为 0(有可能性)，任何更新均不需要。该例程的代码也把上一事件的时间的变量送到当前时间。只要从定时例程返回，主程序就调用该例程，而不管事件类型或者库存水平在这一点实际上是否变化。这提供了一种修改连续时间统计的简单办法(即使计算不是最有效的)。

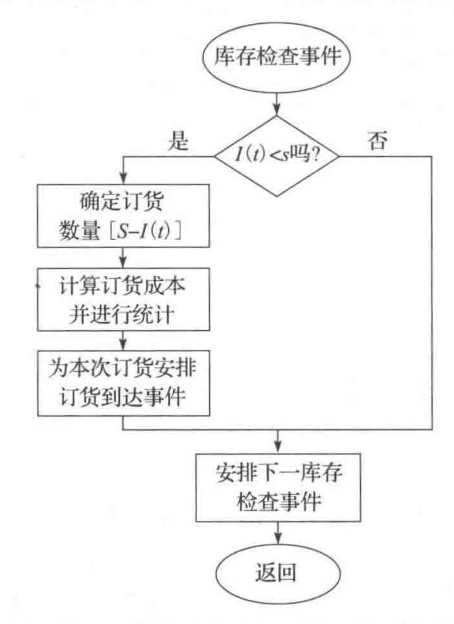


图 1.32　库存检查例程的流程图与库存模型

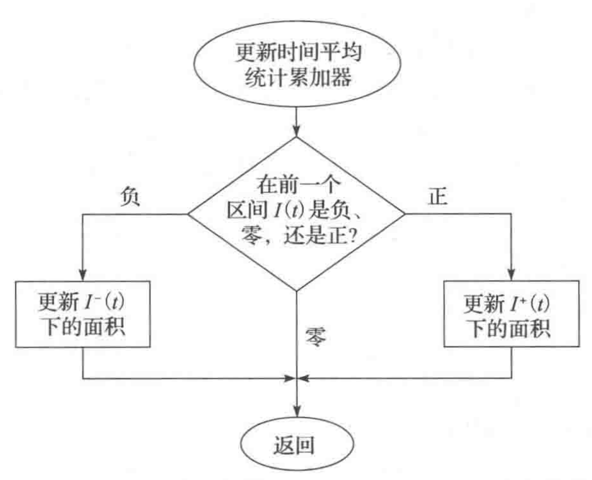


图 1.33　更新连续时间统计累加器的流程图与库存模型

第 1.5.3 小节有对该模型进行仿真的 C 程序。既没有给出定时例程，也没有给出指数变量产生子程序，因为它们与第 1.4 节中的单服务台排队系统模型是完全一样的。读者也应该注意到，排队模型与库存模型的主程序间有相当大的相似性。

1.5.3　C 程序

图 1.34 给出了外部定义。数组 prob_distrib_demand 用来存储需求大小的累积概率，并会传到随机整数发生函数 random _integer()中。如同排队模型一样，我们必须为图 7.5 给出的随机数发生器加入头文件 lcgrand. h(见图 7.6)。所有代码可在地址 www. mhhe. com/law 中找到。

```
/* External definitions for inventory system. */

#include <stdio.h>
#include <math.h>
#include "lcgrand.h"  /* Header file for random-number generator. */

int   amount, bigs, initial_inv_level, inv_level, next_event_type, num_events,
      num_months, num_values_demand, smalls;
float area_holding, area_shortage, holding_cost, incremental_cost, maxlag,
      mean_interdemand, minlag, prob_distrib_demand[26], setup_cost,
      shortage_cost, sim_time, time_last_event, time_next_event[5],
      total_ordering_cost;
FILE  *infile, *outfile;

void  initialize(void);
void  timing(void);
void  order_arrival(void);
void  demand(void);
void  evaluate(void);
void  report(void);
void  update_time_avg_stats(void);
float expon(float mean);
int   random_integer(float prob_distrib []);
float uniform(float a, float b);
```

图 1.34　外部定义的 C 代码与库存模型

图 1.35 给出了主函数的代码。在打开输入、输出文件后，事件数设置为 4。输入参数(除了 s 和 S)于是被读入和写出，并生成报告头；对于每一个(s，S)对，仿真则生成与报告头对应的一行输出的报告函数。然后“for”循环开始，其每一次迭代对于给定的(s，S)对都做一次完整的仿真。模型被初始化，用“do while”保证仿真能持续运转，直到在类型 3 事件发生，如同在第 1.4.6 小节中说的一样。在此循环内部，定时函数用作确定下一个事件类型并更新仿真钟。从定时函数带回下一个事件类型，返回后，连续时间统计在执行事件例程之前更新。然后如以前一样，“switch”语句用做把控制转移到合适的事件例程中。不同于第 1.4.6 小节中的固定时间停止策略，当“do while”循环在此结束时，我们不停止程序，而是进入“for”循环的下一步去读入下一个(s，S)对，再做一个单独的仿真；只有当“for”循环结束，且没有更多的(s，S)对需要考虑时，整个程序才会停止。

```
main()  /* Main function. */
{
    int i, num_policies;

    /* Open input and output files. */

    infile  = fopen("inv.in",  "r");
    outfile = fopen("inv.out", "w");

    /* Specify the number of events for the timing function. */

    num_events = 4;

    /* Read input parameters. */

    fscanf(infile, "%d %d %d %d %f %f %f %f %f %f %f",
           &initial_inv_level, &num_months, &num_policies, &num_values_demand,
           &mean_interdemand, &setup_cost, &incremental_cost, &holding_cost,
           &shortage_cost, &minlag, &maxlag);
    for (i = 1; i <= num_values_demand; ++i)
        fscanf(infile, "%f", &prob_distrib_demand[i]);

    /* Write report heading and input parameters. */
```

图 1.35　主函数的 C 代码与库存模型

```
fprintf(outfile, "Single-product inventory system\n\n");
fprintf(outfile, "Initial inventory level%24d items\n\n",
        initial_inv_level);
fprintf(outfile, "Number of demand sizes%25d\n\n", num_values_demand);
fprintf(outfile, "Distribution function of demand sizes  ");
for (i = 1; i <= num_values_demand; ++i)
    fprintf(outfile, "%8.3f", prob_distrib_demand[i]);
fprintf(outfile, "\n\nMean interdemand time%26.2f\n\n", mean_interdemand);
fprintf(outfile, "Delivery lag range%29.2f to%10.2f months\n\n", minlag,
        maxlag);
fprintf(outfile, "Length of the simulation%23d months\n\n", num_months);
fprintf(outfile, "K =%6.1f   i =%6.1f   h =%6.1f   pi =%6.1f\n\n",
        setup_cost, incremental_cost, holding_cost, shortage_cost);
fprintf(outfile, "Number of policies%29d\n\n", num_policies);
fprintf(outfile, "                 Average        Average");
fprintf(outfile, "         Average        Average\n");
fprintf(outfile, "  Policy       total cost    ordering cost");
fprintf(outfile, "  holding cost   shortage cost");

/* Run the simulation varying the inventory policy. */

for (i = 1; i <= num_policies; ++i) {

    /* Read the inventory policy, and initialize the simulation. */

    fscanf(infile, "%d %d", &smalls, &bigs);
    initialize();

    /* Run the simulation until it terminates after an end-simulation event
       (type 3) occurs. */

    do {

        /* Determine the next event. */

        timing();

        /* Update time-average statistical accumulators. */

        update_time_avg_stats();

        /* Invoke the appropriate event function. */

            switch (next_event_type) {
                case 1:
                    order_arrival();
                    break;
                case 2:
                    demand();
                    break;
                case 4:
                    evaluate();
                    break;
                case 3:
                    report();
                    break;
            }

        /* If the event just executed was not the end-simulation event (type 3),
           continue simulating.  Otherwise, end the simulation for the current
           (s,S) pair and go on to the next pair (if any). */

        } while (next_event_type != 3);
    }

    /* End the simulations. */

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

图 1.35 (续)

图 1.36 给出了初始化函数，我们看到，第一个库存评测安排在时间 0 发生的原因在于，一般说来，初始库存水平会比 s 少。还注意到未考虑事件类型 1(订单到达)，是因为我们的建模假设一开始没有额外的订单。

```
void initialize(void)  /* Initialization function. */
{
    /* Initialize the simulation clock. */

    sim_time = 0.0;

    /* Initialize the state variables. */

    inv_level       = initial_inv_level;
    time_last_event = 0.0;

    /* Initialize the statistical counters. */

    total_ordering_cost = 0.0;
    area_holding        = 0.0;
    area_shortage       = 0.0;

    /* Initialize the event list.  Since no order is outstanding, the order-
       arrival event is eliminated from consideration. */

    time_next_event[1] = 1.0e+30;
    time_next_event[2] = sim_time + expon(mean_interdemand);
    time_next_event[3] = num_months;
    time_next_event[4] = 0.0;
}
```

图 1.36 初始化函数的 C 代码与库存模型

事件函数 order_arrival、demand、evaluate 在图 1.37 到图 1.39 中给出，它们相应于第 1.5.2 小节中给出的一般性讨论及图 1.30 到图 1.32 中的流程图。在 evaluate 中，注意到，对于任何此刻可能发生订货的订单，变量 total_ordering_cost 要增加其订货成本。

```
void order_arrival(void)  /* Order arrival event function. */
{
    /* Increment the inventory level by the amount ordered. */

    inv_level += amount;

    /* Since no order is now outstanding, eliminate the order-arrival event from
       consideration. */

    time_next_event[1] = 1.0e+30;
}
```

图 1.37 order_arrival 函数的 C 代码与库存模型

```
void demand(void)  /* Demand event function. */
{
    /* Decrement the inventory level by a generated demand size. */

    inv_level -= random_integer(prob_distrib_demand);

    /* Schedule the time of the next demand. */

    time_next_event[2] = sim_time + expon(mean_interdemand);
}
```

图 1.38 demand 函数的 C 代码与库存模型

```
void evaluate(void)  /* Inventory-evaluation event function. */
{
    /* Check whether the inventory level is less than smalls. */

    if (inv_level < smalls) {

        /* The inventory level is less than smalls, so place an order for the
           appropriate amount. */

        amount               = bigs - inv_level;
        total_ordering_cost += setup_cost + incremental_cost * amount;

        /* Schedule the arrival of the order. */

        time_next_event[1] = sim_time + uniform(minlag, maxlag);
    }

    /* Regardless of the place-order decision, schedule the next inventory
       evaluation. */

    time_next_event[4] = sim_time + 1.0;
}
```

图 1.39 evaluate 函数的 C 代码与库存模型

报告生成器在图 1.40 列出，它分别计算了总成本的三个组成部分，并将它们加在一起，得到月平均总成本。为了区分，s 和 S 的当前值会与平均总成本及其三个组成成分(订货，储备，和缺货成本)一并写出。

```
void report(void)  /* Report generator function. */
{
    /* Compute and write estimates of desired measures of performance. */

    float avg_holding_cost, avg_ordering_cost, avg_shortage_cost;

    avg_ordering_cost = total_ordering_cost / num_months;
    avg_holding_cost  = holding_cost * area_holding / num_months;
    avg_shortage_cost = shortage_cost * area_shortage / num_months;
    fprintf(outfile, "\n\n(%3d,%3d)%15.2f%15.2f%15.2f%15.2f",
            smalls, bigs,
            avg_ordering_cost + avg_holding_cost + avg_shortage_cost,
            avg_ordering_cost, avg_holding_cost, avg_shortage_cost);
}
```

图 1.40 report 函数的 C 代码与库存模型

函数 update_time_avg_stats 在第 1.5.2 小节和图 1.33 中的流程图中进行了一般性讨论，并在图 1.41 给出。注意，如果库存水平 inv_level 为 0，即“if”和“else if”的都不满足时，如所要求的，将没有任何更新发生。如同在第 1.4 节中的单服务台排队模型那样，如果仿真运行时间很长，那么必须在该例程的顶部把变量 sim_time 和 time_last_event 都定义为 double 类型，以避免做减法时造成四舍五入错误。

函数 random_integer 在图 1.42 中给出，它的实现没有什么特别，会根据分布函数 prob_distrib[I]产生整数，prob_distrib[I]中的值是给定的(在我们的例子中，prob_distrb[1]=1/6，prob_distrb[1]=1/2，prob_distrb[1]=5/6，prob_distrb[1]=1，所有给定值在输入时保留十进制小数三位精度)。其逻辑与第 1.5.2 小节中的讨论是一致的；注意，输入数组 prob_distrib 必须包含累积分布函数而不是变量取某可能值的概率。

函数 uniform 在图 1.43 中给出，并在第 1.5.3 小节中进行了描述。

```
void update_time_avg_stats(void)  /* Update area accumulators for time-average
                                     statistics. */
{
    float time_since_last_event;

    /* Compute time since last event, and update last-event-time marker. */

    time_since_last_event = sim_time - time_last_event;
    time_last_event       = sim_time;

    /* Determine the status of the inventory level during the previous interval.
       If the inventory level during the previous interval was negative, update
       area_shortage.  If it was positive, update area_holding.  If it was zero,
       no update is needed. */

    if (inv_level < 0)
        area_shortage -= inv_level * time_since_last_event;
    else if (inv_level > 0)
        area_holding  += inv_level * time_since_last_event;
}
```

图 1.41 函数 update_time_avg_stats 的 C 代码与库存模型

```
int random_integer(float prob_distrib[])  /* Random integer generation
                                             function. */
{
    int   i;
    float u;

    /* Generate a U(0,1) random variate. */

    u = lcgrand(1);

    /* Return a random integer in accordance with the (cumulative) distribution
       function prob_distrib. */

    for (i = 1; u >= prob_distrib[i]; ++i)
        ;
    return i;
}
```

图 1.42 函数 random_integer 的 C 代码

```
float uniform(float a, float b)  /* Uniform variate generation function. */
{
    /* Return a U(a,b) random variate. */

    return a + lcgrand(1) * (b - a);
}
```

图 1.43 uniform 函数的 C 代码

1.5.4 仿真输出和讨论

仿真报告(在文件 inv. out 中)在图 1.44 中给出。对本模型来说，即使使用同一随机数发生器算法，但不同的编译器和计算机给出的结果可能会有一些区别；这个区别可见第 1.4.4 小节开头的一些讨论。

在报告中给出了月平均总成本的三个不同组成部分以便观察它们各自如何随 s 和 S 变化而变化的情况，这是作为对模型和代码的一个可能的检查。例如，固定 $s=20$，且 S 从 40 增长到 100，使存储成本每个月从 \$9.25 稳定增长到 \$36，而同时使缺货成本减少；S

```
Single-product inventory system

Initial inventory level                     60 items

Number of demand sizes                       4

Distribution function of demand sizes     0.167   0.500   0.833   1.000

Mean interdemand time                       0.10 months

Delivery lag range                          0.50 to      1.00 months

Length of the simulation                    120 months

K =  32.0    i =   3.0    h =   1.0    pi =   5.0

Number of policies                           9
```

Policy	Average total cost	Average ordering cost	Average holding cost	Average shortage cost
(20, 40)	126.61	99.26	9.25	18.10
(20, 60)	122.74	90.52	17.39	14.83
(20, 80)	123.86	87.36	26.24	10.26
(20,100)	125.32	81.37	36.00	7.95
(40, 60)	126.37	98.43	25.99	1.95
(40, 80)	125.46	88.40	35.92	1.14
(40,100)	132.34	84.62	46.42	1.30
(60, 80)	150.02	105.69	44.02	0.31
(60,100)	143.20	89.05	53.91	0.24

图 1.44 输出报告与库存模型

的这种增加对订货成本的影响则是减少了订货成本，最终，将订货提高到更大的 S 意味着订单越大，订货发生的频率就越低，这样避免了的固定订货成本的高频率发生。类似地，例如，将 S 的值固定在 100，那么 s 从 20 增加到 60 导致缺货成本减小（\$7.95，\$1.30，\$0.24），但存储成本增加（\$36.00，¥46.42，\$53.91），因为 s 的增加转化为不希望库存水平降低到一个低值。虽然不做仿真我们有可能预测成本的这些组成部分的变化趋势，但关于它们的变化幅度在没有仿真的帮助下不可能说出太多的东西来。

由于总准则月总成本是三部分成本的总和，而在某些时候，三部分成本对 s 和 S 变化的反应是以不同的方向移动的，所以没有仿真帮助，我们无法预测该准则的移动方向。所以，我们只要看该准则的值，从而发现(20，60)这个策略应该是最好的，其月平均总成本为 \$122.74。然而，在目前的问题中，仿真长度固定（公司计划的时间为 10 年），我们真正想要估计的是最初 120 个月的期望月平均总成本。图 1.44 所示的数是这些期望值的估计，每一个估计都是基于大小为 1 的样本（仿真运行或重复运行）。由于这些估计可能有很大的方差，它们的顺序可能与期望值的顺序有很大的不同，后者是想要的信息。实际上，如果我们使用不同的 $U(0,1)$ 随机变量来重新运行 9 次仿真，得到的估计值可能与图 1.44 所示的有很大不同，而且，新估计值的顺序也可能不同。

从上面的讨论我们得出结论，当问题要求仿真运行长度固定时，对于每一个策略或感兴趣的系统只运行一次仿真，一般来说是不够的。在第 9 章我们将讨论到底需要多少次仿真运行才能为所要求的期望值得到一个好的估计。涉及由不同系统设计而产生几个不同期

望值的问题，我们将在第 10 章到第 12 章讨论。

1.6　并行/分布式仿真和高层体系结构

第 1.4 和第 1.5 节中的仿真(以及第 2 章所考虑的那些仿真)基本上都以同样的方式运行。仿真钟和事件表相互作用以确定哪一个会是下一个被处理事件，仿真钟推进到这个事件的发生时间，计算机处理该事件的逻辑，其中可能包括更新状态变量，更新事件列表和收集统计数据。该逻辑按事件发生的仿真时间顺序来执行，即仿真是“串行的”(sequential)。另外，所有工作都是在单台计算机上完成的。

近些年来，计算机技术已经可以使单处理器或计算机连接起来，形成并行或分布式计算环境。这就使得多处理器同时处理一个仿真模型的不同部分成为可能，从而减少了完成仿真的总时间。或者，运行在不同计算机上的两个或者多个不同仿真模型由网络联系在一起形成一个整体的仿真模型，其中每个模型与其他模型随时间交互效用。本节我们介绍这样的备选方法以执行一个仿真模型。

1.6.1　并行仿真

并行离散事件仿真[见 Fujimoto(1998, 2000, 2003)]涉及在紧耦合的计算机系统上(例如，一个超级计算机或者一个共享内存的多处理器)仿真模型的执行。通过将仿真的执行分散到几个不同的处理器，期望模型执行时间能明显地减少(甚至达到处理器数分之一)。举例来说，如果有人仿真具有上千个节点的通信网络或者很大的军事模型，那么执行时间会非常长，从而需要考虑使用并行仿真。并行仿真的另外一个可能的应用是实时决策。例如，在空中交通控制系统中，人们需要对几个小时的空中交通进行仿真以决定“当前”最好如何重新规划交通[见 Wieland(1998)]。

为了开发并行仿真，一个模型需要分解成几个逻辑过程(LP)(或子模型)。每个 LP(或每组 LP)分配到不同的处理器中，每一个处理仿真模型的一部分。通过互相发送有时间标记的信息或事件，LP 之间互相通信。举例来说，一个制造系统的建模典型的做法是将其建成为相互连接的排队系统的网络，每一排队系统表示一个不同的工作站。当一个作业离开某工作站时，一个“到达”事件必须发送到该作业路径上的下一站(除非此作业已离开该系统)。

并行仿真中一个至关重要的问题是保证整个仿真模型中的事件，无论它们的 LP 是什么，都要按其合适的时间顺序进行处理。举例来说，假设在发生一个特定的作业到达一个工作站之后，另一个作业从不同的工作站离去，那么必须有一个同步机制来确保是这样发生的。如果每个 LP 以事件时间的增序来处理其所有事件(或者是它自己产生的，或者是其他 LP 产生的)，一种称为局部因果约束的要求，那么可以证明，并行仿真产生的结果就会与整个模型在单台计算机上串行运行完全相同。

每一个 LP 可以看做是串行的离散事件仿真模型，拥有其自身的局部状态变量，事件表和仿真钟。然而，整个并行仿真模型并没有与串行仿真模型的情况那样的全局类似模型。

从历史上看，采用两种不同类型的同步机制：保守的和乐观的。在保守同步机制中[见 Bryant(1977)和 Chandy and Misra(1979)]，目标是绝对避免违反局部因果约束。例如，假设一个特定的 LP 目前在仿真时间为 25，并为事件时间为 30 的下一事件的处理做好了准备。那么，时间同步机制必须保证此 LP 以后不会接收到来自另一个 LP 的事件时间小于 30 的事件。因而，此机制的目标是要确定何时处理一个特定的事件才是真正“安全”的，即要确定何时能保证此 LP 以后不会收到具有更小事件时间的事件。

保守同步机制有如下两个缺点[见 Fujimoto(1998, 第 444-446 页)]。

(1)它们不能充分利用在仿真应用中可用的并行性。如果事件 A 可能以某种形式影响事件 B，那么 A 和 B 必须按顺序执行。如果在仿真模型是 A 很少影响 B 这种情况，那么

A 和 B 在大多数时间可以同时处理。

(2)鲁棒性不是很好——模型中一个看起来很小的变化可能会导致其性能严重的下降。

在乐观同步机制中，违反局部因果约束的情况允许发生，但同步机制检测违反，会从违反恢复。如上所述，每一个 LP 按时向前仿真其自己的那一部分模型，但不等待接受来自具有不同推进速率的其他处理器的信息——这种等待对保守同步是必需的。

时间回滚机制[time-warp mechanism，见 Jefferson(1985)]是最流行的乐观同步方法。如果一个 LP 收到了一个本该在之前就应收到的信息(那么，从本该收到的那一点起可能就会对其动作产生影响)，于是这个接收 LP 发生回滚，由此其仿真时钟恢复到(早先的)信息到达的时间。举例来说，如果 LP A 仿真到时间为 50 时，一条来自 LP B 的信息到达，然而此信息本应在时间 40 时到达，则 A 的时钟会回滚至 40，A 在 40 到 50 之间的仿真被取消，因为没有 40 时刻信息的内容，仿真有可能不正确。部分被取消的工作可能已经给其他的 LP 发送了信息，那么发送一个反消息使每一个已经发出的信息无效——这些反消息在其目的地 LP 会自身生成第二级回滚，依此类推。

乐观同步机制可以比保守同步机制更好地利用仿真应用中的并行性，这是由于它们不受最坏情况限止(见保守同步机制的缺点(1))。然而，它们仍有自己的缺点[见 Fujimoto (1998，第 448-451 页)]：

(1)它们执行回滚引起额外的计算。

(2)它们需要更多的计算机内存，因为每一个 LP 的状态必须定时保存，以便从回滚中复原。

上文介绍的空中交通控制仿真应用中就使用了乐观同步机制，并运行在一个四处理器、共享内存的 Sun 工作站上。

多年来，由于计算机处理器的处理速度每 18 个月就翻倍，需要并行仿真技术使执行时间在可接受的时间内的仿真模型的数量明显地越来越少。另一方面，并行仿真可以使不适合单机内存的某些模型能够运行。例如，Fujimoto 等在 2003 年展示了并行仿真方法可以使被仿真的通信网络的规模得到惊人的扩大。

1.6.2 分布式仿真和高层体系结构

分布式仿真最初用于建立一个完整仿真模型，该模型由两个或多个位于联网的计算机上的单独模型组成。这种分布式仿真的兴趣起源于希望建立实时、人在回路的仿真以便能用于训练军队人员。从 1983 年到 1990 年的 SIMNET(SIMulator NETworking)项目演示了此概念的可行性。这导致一组仿真互联协议诞生，这就是众所周知的分布式交互仿真(DIS)标准。DIS 后来又让位于高层体系结构(HLA)[参见 Dahmann 等(1998)，Kuhl 等(2000)，和网站 www. hla. dmso. mil]，它是在美国国防部建模与仿真办公室领导下由美国国防部开发的。

HLA(IEEE 标准 1516)是为推动仿真重用和互操作而设计的软件架构。它是基于没有一种仿真可以满足国防工业的所有用途和应用的一个假设，而且它最终会降低为了建立新的目标综合环境所需要的时间和成本。HLA 可以将国防部定义的下列仿真类型组合在一起：

- 实况仿真——真人操作实际系统(例如，现场测试)；
- 虚拟仿真——真人操作仿真系统(例如，坦克驾驶舱仿真器里的人与仿真生成的敌人兵力作战)；
- 构造仿真——仿真的人操作仿真的系统(例如，离散事件仿真)。

所有美国国防部仿真均假定为从 2001 年 1 月 1 日起与 HLA 兼容，除非得到特别允许。

一个 HLA 的联邦由一组相互作用的、称为联邦成员的仿真个体，运行支撑环境(RTI)，以及接口组成，如图 1.45 所示。RTI 提供了一组通用服务，支持仿真进行联邦

成员对联邦成员的交互，它还提供联邦成员管理的功能。联邦成员间的所有交互都要通过RTI，RTI的软件和算法并不由HLA定义(RTI的软件可以从第三方供应商买到)。HLA运行期接口规范为联邦成员与RTI交互、调用RTI服务以支持联邦成员间的交互，以及响应来自RTI的请求提供一个标准机制。接口是独立实现的，从而不依赖"对象"模型(例如，对于一个实体)和任何联邦成员的数据交换的需要[HLA对象并没有像经典的面向对象的仿真那样的方法(见第3.6节)]。

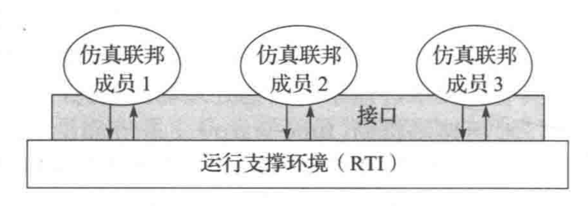


图1.45 HLA联邦的功能视图

HLA由三个部分正式定义：接口规范、对象模型模板和规则。HLA接口规范描述了通过RTI提供给联邦成员和通过联邦成员提供给RTI的运行服务。有六类服务提供创建和操作联邦的能力，时间管理(即同步)的能力和联邦执行时联邦成员间有效的数据分发能力等。

HLA对象模型是以对象的方式对联邦的本质上可共享的元素的描述。由于HLA是面向互操作的，对象模型描述了在整体仿真模型中共享的联邦成员和联邦的关键方面。HLA对对象模型的内容没有约束，但需要这些模型以所谓对象模型模板(OMT)的标准格式形成文件。有两类对象模型：联邦对象模型(FOM)和仿真对象模型(SOM)。HLA FOM描述了联邦间共享的对象、属性和相互作用(例如事件)集。HLA SOM用对象、属性和相互作用的方式描述一个仿真(联邦)，它可以供以后的联邦使用，这使得评价一个仿真是否适合加入一个新的联邦变得容易。

HLA规则总结了HLA底层的关键原则，并可分为两组：联邦规则与联邦成员规则。联邦规则规定每一个联邦必须有一个FOM，这样，一个联邦中的所有对象描述都保留在联邦成员里而不是在RTI里，等等。联邦成员规则声明了仿真的公共信息在SOM文件里，这样使用RTI提供的时间管理服务就可以完成本地时间管理，等等[关于HLA中时间管理的讨论请参见Fujimoto(2003)]。

国防部的HLA联邦用于训练军人，测试与评估部队装备，以及分析新军事系统和策略。后一类应用的一个例子是美国海军使用HLA构建网络模拟战(NETWARS)和海军仿真系统(NSS)联邦[参见Alspaugh等(2004)和Murphy与Flournoy(2002)]。此联邦采用保守时间管理。

除了国防领域，对分布式仿真和HLA感兴趣的人也大有人在。例如，美国国家标准与技术局(NIST)的MISSION项目把HLA应用于多机构(例如，一个供应商和一个运输公司)供应链的分布式仿真中[参见McLean与Riddick(2000)]。由于一个机构不希望别的供应链机构知道它的运转细节，所以分布式仿真可能是必需的。

虽然HLA最初是为将两个或多个独立仿真形成联邦而设计，但是现在也用于将同一仿真模型进行多个拷贝形成自联邦。例如，Fujimoto等(2003年)，Bodoh和Wieland(2003年)分别将这种方法应用到大型通信网络和商业空中交通控制的并行仿真模型中。

分布式仿真的一个完全不同的用途是在连网计算机上进行单机仿真模型的独立重复运行。这将使分析人员在给定的"墙钟"时间对一个特定的系统配置进行多个重复运行成为可能，其结果是感兴趣的性能度量的统计估计更精确。这也使得分析人员在试图"优化"一个感兴趣的系统的性能时，在给定的"墙钟"时间(wall-clock time)里，能对大量的不同的系统配置进行仿真(参见第12.5节)。仿真包AutoMod(见第13章)明确支持分布式仿真的这种用法。

关于并行和分布式仿真的其他信息可以在期刊"*ACM Transactions on Modeling and Computer Simulation*"(TOMACS)中找到，也可以从年刊"*Proceedings of the Winter*

Simulation Conference" 和 "*Proceedings of the Workshop on Parallel and Distributed Simulation*"中找到。

1.7 一个有效的仿真研究的步骤

现在我们稍微详细地看看离散事件仿真的内部工作。我们需要回头并认识到，用仿真来设计或分析一个复杂系统，模型编程只是整个工作的一部分。必须将注意力放到很多其他相关的方面，诸如对系统随机性建模、有效性、仿真输出数据的统计分析和项目管理。图1.46给出了一个典型的、有效的仿真研究的各个步骤[也可参见Banks等(2005，第14-18页)和Law(2003)]。表示每一步的符号旁边的数字作为后面该步的详细说明的参考。注意，仿真研究并不是一个简单的串行过程。人们随着研究的推进，也许必须返回到前面的某一步。

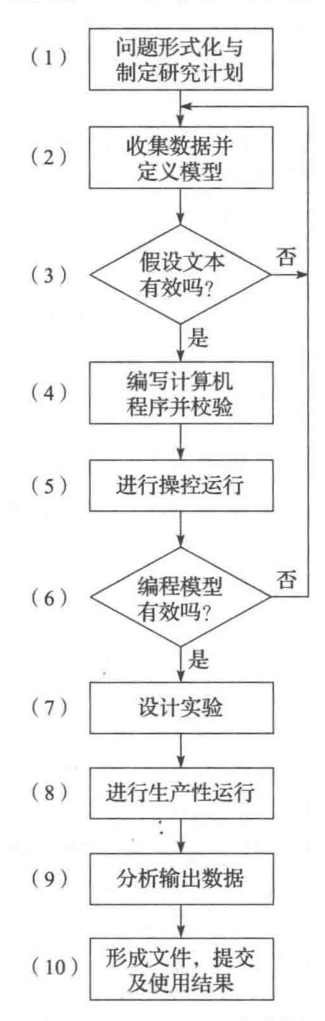


图1.46 仿真研究的步骤

(1)问题形式化与制定研究计划。

a. 项目经理说明感兴趣的问题。

- 问题可能没有正确地说明或没有以定量的方式说明。
- 往往需要迭代过程。

b. 召开一个或多个本研究的启动会议，包括项目经理、仿真分析员和负责的领域专家(SME)，讨论下列事情：

- 本研究的总目标；
- 通过该研究要回答的特殊问题(用于决定模型的详细程度)；
- 为评价不同系统配置的功效将采用的性能度量；
- 模型的范围；
- 被建模的系统配置(用于决定仿真程序的通用性)；
- 本研究的时间框架和需要的资源。

(2)收集数据并定义模型。

a. 收集有关系统结构和操作程序的信息。

- 一个人或一个文件是不充分的。
- 某些人的信息可能不准确——确保找到真正的领域专家。
- 操作程序可能未正式形成。

b. 收集数据(如果可能)以确定模型参数和输入概率分布(参见第6章)。

c. 用书面的假设文件描述上述的信息和数据(参见第5.4.3小节)。

d. 收集有关现有系统性能的数据(如果可能)(为了第6步的有效性确认目的)。

e. 确定模型的详细程度(参见第5.2节)，这是一门艺术，取决于下面的内容：

- 项目目标；
- 性能度量；
- 数据有效性；
- 可信性的考虑；
- 计算机的限制；
- 领域专家的意见；
- 时间和金钱的限制。

f. 模型中的元素和对应系统中的元素并不应该是一一对应的。

g. 从一个“简单”的模型开始，并根据需要去完善它。为使决策有效，很少需要对系统中的每个方面都要建模，否则会导致模型执行时间过长，错过了截止日期，或掩盖了重要的系统因素。

h. 定期和项目经理(和其他关键项目人员)互动(见第 5.4.2 小节)。

(3)假设文档是有效的吗?

在项目经理、分析师和领域专家面前结构性地浏览这些假设文档(参见第 5.4.3 小节)，这可以：

- 有助于确保模型假设是正确而完整的；
- 促进项目成员之间的互动；
- 明确模型的归属；
- 在编程开始之前做，以免日后大量的重复编程。

(4)编制计算机程序并校验。

a. 用编程语言(例如，C 或 C++ 语言)或者仿真软件(例如，Arena、Extend、Flexsim 和 ProModel)对此模型编程。使用编程语言的好处是众所周知的，这些好处是提供更大的程序控制，购买费用低，模型执行的时间短。从另一方面说，使用仿真软件(参见第 3 章)可以减少编程时间并降低项目成本。

b. 校验(调试)仿真计算机程序(参见第 5.3 节)。

(5)进行操控运行。

为了第(6)步中的有效性确认目的，进行操控运行。

(6)编程模型有效吗?

a. 如果现有一个系统，那么将模型与现有系统的系统性能(来自第(2)步)度量进行比较。

b. 无论是否有一个实际系统，仿真分析家和领域专家应当对模型结果的正确性进行评价。

c. 使用灵敏度分析(见第 5.4.4 小节)确定哪个模型因素对性能度量有明显的影响，从而这些因素的建模必须仔细。

(7)设计实验。

为感兴趣的每一个系统配置，详细说明以下事项：

- 每次仿真运行的长度；
- 如果合适的话，预热期的长度；
- 使用不同随机数运行的独立仿真的次数(参见第 7 章)——帮助构建置信区间。

(8)进行批次运行。

进行批次运行用于第(9)步。

(9)分析输出数据。

分析输出数据的两个主要目标是：

- 确定某些系统配置的绝对性能(参见第 9 章)；
- 在相对意义下比较不同的系统配置(参见第 10 章和第 11.2 节)。

(10)形成文件，提交及使用结果。

a. 为了目前的和未来的项目的使用，对假设(见第(2)步)，计算机程序和研究结果形成文件。

b. 提交研究结果。

- 使用动画(参见第 3.4.3 小节)向项目经理以及其他不熟悉所有模型细节的人介绍模型。
- 为了增强可信性，讨论模型的建立和确认过程。

- 如果结果都是有效并可信的，那么在决策过程中使用该结果。

1.8 仿真的优点、缺点和缺陷

我们通过列出仿真的一些好的和不好的特性(相对于其他研究系统的方法而言)和说明在仿真研究中能对仿真项目造成损害甚至破坏的一些常见错误，来结束本章。这个主题在第 1.2 节中也进行了一定程度上的讨论，但我们现在会通过一些仿真例子来讨论，这可能会更专业。

正如在第 1.2 节中提到的，对研究复杂系统来说，仿真是一个广泛使用并迅速流行的方法。仿真的广泛使用归因于其具有如下某些可能的优点。

- 大多数复杂的、真实的具有随机因素的系统不能用可解析求解的数学模型来精确描述，于是，仿真研究往往是唯一可能的方法。
- 在某些计划的一组运行条件下仿真可使人们估计一个现有系统的性能。
- 通过仿真可以比较各种建议的系统设计(或一个系统的各种运行策略)，以观察哪一个最好地满足了规定的要求。
- 在仿真中我们可以更好地控制实验条件，这些实验条件在用真实系统本身做试验时中往往可能出现。
- 仿真可以让我们以压缩时间的办法来研究一个具有长期规划的系统，例如，经济系统；或另一种，以扩展时间的办法来研究系统的详细工作情况。

仿真不是没有其缺点，其某些缺点如下。

- 随机仿真模型的每一次运行产生的只是在特定的一组输入参数下的模型真实特征的一个估计。这样，对于每一组所研究的参数，恐怕需要多次独立地运行模型(见第 9 章)。由于这个原因，给定的多种系统设计进行固定次数比较，仿真模型一般不是最优的。另一方面，如果适合的话，解析模型在不同的输入参数集下往往易于得到模型的准确的真实特性。因此，如果一个“有效的”解析模型是可用的或能易于开发出来，它一般比仿真模型会更可取。
- 开发仿真模型通常费钱费时。
- 仿真研究中产生的大量数据或者逼真动画的有说服力的影响往往造成一种倾向，认为研究结果的置信度大于证明的结果。如果模型不是对系统“有效”的表达，那么无论仿真结果看起来多么让人印象深刻，对真实系统来说，它们提供的信息是没有多大用处的。

当在给定的环境下决定仿真研究是否合适时，我们只能建议记住这些优点和缺点，同时还要注意这些特殊环境的所有其他相关方面。最后，请注意，在某些研究中，仿真和解析模型都是有用的。特别地，仿真可以用来检验解析模型中所需假设的有效性。另一方面，解析模型可以为仿真研究提供更合理的研究方法。

假设我们决定使用仿真，我们发现下列缺陷会影响仿真研究成功完成[也可见 Law 和 McComas(1989)]：

- 在仿真研究的开始阶段没有很好地定义目标集。
- 没有让整个项目成员从一开始就都参与研究。
- 模型详细程度不合适。
- 在整个仿真研究的过程中缺乏和管理交流。
- 管理层对仿真的错误理解。
- 将仿真研究视为它主要是计算机编程练习。
- 建模团队缺乏具有仿真方法学和统计学知识的人(第 5 章、第 6 章、第 9 章等)。
- 没有收集到良好的系统数据。
- 不合适的仿真软件。

- 使用一些仿真软件产品，忽视了它们的复杂的宏语句可能不便于形成文件，而且也许未实现期望的建模逻辑。
- 相信易用的仿真包，它只需要很少或不需要编程，需要的技术力量明显降低。
- 滥用动画。
- 未正确地考虑到真实系统中随机性来源。
- 使用任意的分布(例如，正态分布，均匀分布，或三角形分布)作为仿真的输入。
- 使用基于独立性假设的公式，由一次仿真运行(重复运行)分析输出数据。
- 对特定的系统设计只做了一次重复运行，就把输出统计当作“真实答案”。
- 如果关心的是系统的稳态性能，却没有预热期。
- 对每个设计，基于一次重复运行来比较不同的系统设计。
- 使用了错误的性能度量。

在本书剩下的章节中我们会更多地说明“做什么”(而不是“不做什么”)。

附录 1A　固定增量时间推进

如同在第 1.3.1 小节中提到的，在离散事件仿真模型中推进仿真钟的第二种基本方法叫固定增量时间推进。采用这种方法，适当选择 Δt，仿真钟推进的增量正好是 Δt 时间单位。在仿真钟每更新一次后，会做一个检查，以确定在前一个 Δt 区间长度中是否有事件发生。如果有一个或更多的事件安排在此区间内发生，那么就认为这些事件是在此区间的终点发生，其系统状态(和统计计数器)做相应的更新。固定增量时间推进方法在图 1.47 中给出，其中带箭头的曲线表示仿真钟的推进，而 $e_i(i=1, 2, \cdots)$是任意类型的第 i 个事件发生的真实时间(不是仿真钟的第 i 个值)。在时间区间$[0, \Delta t)$中，一个事件在时间 e_1 发生，但模型认为是在 Δt 时刻发生。在$[\Delta t, 2\Delta t)$中没有事件发生，但模型进行检查以确定确实如此。在区间$[2\Delta t, 3\Delta t)$中的 e_2，e_3 时刻发生了事件，但两事件都被认为是在 $3\Delta t$ 时刻发生的，依此类推。当两个或更多事件被模型认为是同时发生时，必须为此模型建立一组规则，以决定用何种顺序来处理这些事件。固定增量时间推进方法的两个缺点是，在事件发生区间的终点处理事件会引入误差，以及当实际不是同时的事件被模型处理为同时事件时，需要决定先处理哪个事件。通过使 Δt 变得小一些，这些问题可以变得不那么严重，但这增加了检查事件发生的次数(这是必须要做的)，从而导致执行时间的增加。由于这些原因，当连续事件间的时间可能变化很大时，固定增量时间推进一般不用于离散事件仿真模型。

此方法主要应用于这样的系统：适当选择 Δt，可以合理地假设所有事件实际上发生在 $n\Delta t(n=0, 1, 2, \cdots)$的某一个时刻。例如，经济系统中的数据往往只是按年度为基础才是可用的，所以仿真模型很自然就以 1 年为增量来推进仿真钟[参见 Naylor(1971)对经济系统仿真的讨论，也可参见第 1.8.4 小节库存系统的讨论，该系统能采用固定增量时间推进方法进行仿真，没有精度损失]。

注意，使用图 1.47 所示下一事件时间推进方法可以实现固定增量时间推进，办法是在每个 Δt 时间单位，人为地安排“事件”发生。

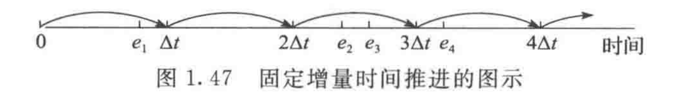


图 1.47　固定增量时间推进的图示

附录 1B　排队系统入门

排队系统由一个或多个为到达顾客提供某一类服务的服务台组成。客户到达发现所有服务台都在忙(一般情况下是这样)时会加入服务台前的一个或多个队列(或线)，因而取名“排队”系统。

从历史上看，大部分的离散事件仿真研究都包含了实际排队系统的建模，或至少是这些被仿真的系统的某些组成部分为排队系统。因此，我们认为对于学习仿真的学生来说，对排队系统的成分，排队系统的标准符号，以及经常用于指示由排队系统提供的服务的质量的性能度量至少有一个基本的了解是很重要的。表 1.4 给出了一些经常被仿真的实际排队系统的一些例子。如果想了解更多关于一般排队系统的信息，请见 Gross 和 Harris (1998)。对通信网络排队模型感兴趣的同学建议参考 Bertsekas 和 Gallager(1992)的书。最后，Shanthikumar 和 Buzacott(1993)讨论了制造系统的随机模型。

表 1.4 排队系统的例子

系统	服务台	顾客
银行	出纳台	顾客
医院	医生，护士，床位	病人
计算机系统	中央处理单元，输入/输出设备	作业
制造系统	机器，工人	零件
机场	跑道，门，安检站	飞机，旅行者
通讯网络	节点，链路	信息，包

1B.1 排队系统的组成

排队系统由三部分组成：到达过程，服务机制和排队规则。排队系统到达过程的说明包括描述客户如何到达此系统的。令 A_i 为第 $i-1$ 和第 i 个顾客到达之间的到达间隔时间(参见第 1.3 节)。如果假设 A_1，A_2，…为独立同分布的随机变量，我们将用 $E(A)$来表示平均(或期望)到达间隔时间，并称 $\lambda=1/E(A)$是顾客的到达速率。

排队系统的服务机制描述包括说明服务台的数量(用 s 表示)，是每一个服务台有其自己的队列，还是只有一个队列进入所有的服务台，以及顾客服务时间的概率分布。令 S_i 表示第 i 个到达顾客的服务时间。如果 S_1，S_2，…是独立同分布的随机变量，我们将用 $E(S)$表示顾客的平均服务时间，并称 $\omega=1/E(S)$为服务台的服务速率。

排队系统的排队规则指的是服务台在完成当前顾客的服务后，从队列中选择下一个顾客(如果有的话)所采用的规则。通常使用的排队规则如下。

先进先出(FIFO)：以先进先出方式服务顾客。

后进先出(LIFO)：以后进先出方式服务顾客(见习题 2.17)。

优先级：按顾客的重要性(见习题 2.22)或基于顾客的服务需求(见习题 1.24，习题 2.20 和习题 2.21)服务顾客。

1B.2 排队系统的表示符号

某些排队系统如此频繁地出现在实际中，以致已经形成了它们的标准符号。特别是，考虑图 1.48 所示的排队系统，有以下特征：

(1) s 个并行的服务台和一个进入所有服务台的先入先出队列。

(2) A_1，A_2，…是独立同分布的随机变量。

(3) S_1，S_2，…是独立同分布的随机变量。

(4)所有 A_i 和所有 S_i 都是独立的。

我们把这样的系统叫做 $GI/G/s$ 队列，其中 GI(general independent，一般独立)指的是 A_i 的分布，G(general)指的是 S_i 的分布。如果 A_i 和 S_i 是特定的分布(在仿真中经常是这种情况)，那么表示这些分布的符号应该在 GI 和 G 的位置标出。由于指数分布具有马尔可夫(Markov)性质，即无记忆性质，所以我们用符号 M 表示指数分布(见习题 4.26)，符号 E_k表示 k -厄兰(Erlang)分布(如果 X 是一个 k -厄兰随机变量，那么 $X=\sum_{i=1}^{k}Y_i$，其中

Y_i 是独立同分布的指数随机变量），用 D 表示确定（或常量）时间。这样，具有指数到达间隔时间和服务时间和先入先出排队规则的单服务台排队系统称为 $M/M/1$ 队列。

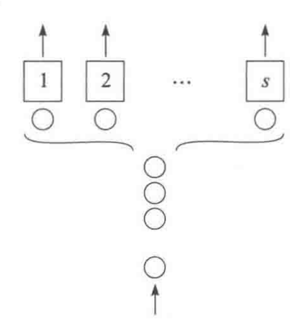


图 1.48 $GI/G/s$ 队列

对于任意 $GI/G/s$ 队列，我们把 $\rho=\lambda/(s\omega)$ 这个量叫做排队系统的利用率（$s\omega$ 是所有服务台都在忙时的系统服务速率）。它是排队系统资源利用程度如何的一个度量。

1B.3 对排队系统性能的评价

排队系统有很多可能的性能度量。这里，我们介绍在用数学方法研究排队系统中四种常用的性能度量。读者不应从我们的选择中推断出这些度量必然是在实际中最相关或最重要的（更多讨论请见第 9 章）。实际上，对于一些实际系统，这些度量可能根本不好定义，即它们可能并不存在。

令：

D_i 为第 i 个顾客在队列中的延误时间；

$W_i=D_i+S_i$ 为第 i 个顾客在系统中的等待时间；

$Q(t)$ 为在 t 时刻队列中的顾客数；

$L(t)$ 为在 t 时刻系统中的顾客数［$Q(t)$ 加上在 t 时刻正在接受服务的顾客数］。

于是度量

$$d=\lim_{n\to+\infty}\frac{\sum_{i=1}^{n}D_i}{n},\quad \text{w. p. 1}$$

和

$$w=\lim_{n\to+\infty}\frac{\sum_{i=1}^{n}W_i}{n},\quad \text{w. p. 1}$$

（如果它们存在）称为稳态平均延误时间和稳态平均等待时间。类似地，度量

$$Q=\lim_{T\to+\infty}\frac{\int_0^T Q(t)\,dt}{T},\quad \text{w. p. 1}$$

和

$$L=\lim_{T\to+\infty}\frac{\int_0^T L(t)\,dt}{T},\quad \text{w. p. 1}$$

（如果它们存在）称为稳态时间平均队列人数和稳态时间平均系统人数。限制符“w. p. 1”（具有 1 的概率）在此和在本书中，是为了数学精确性而给出，并不具有太多实际意义。例如，假设一些排队系统当 $n\to+\infty$ 时，$\sum_{i=1}^{n}D_i/n\to d$ (w. p. 1)。这意味着如果进行大量（无穷多个）实验，那么实质上每个实验的 $\sum_{i=1}^{n}D_i/n$ 值都收敛于 d。注意，$\rho<1$ 是 $GI/G/s$ 队列存在 d，w，Q，L 的必要条件。

排队系统最普遍而有用的结果是守恒方程

$$Q=\lambda d \quad 和 \quad L=\lambda w$$

对于存在 d 和 w 的所有排队系统，这两个方程成立［参见 Stidham(1974)］。第 11.5 节给出了对这些关系的仿真应用。另一个有重要实际价值的等式是：

$$w=d+E(S)$$

更多讨论请见第 1.4.5 小节和第 11.5 节。

值得一提的是，上述讨论的性能度量，对于 $M/M/s$ 队列（$s\geqslant1$）、任意 G 分布的 $M/$

$G/1$ 队列，以及某些其他排队系统，是可以解析计算出来的。一般说来，到达间隔时间分布，服务时间分布，或者两者必须是指数分布(或指数分布的变种，例如 k 维厄兰分布)，使得解析求解成为可能[参见 Gross 和 Harris(1998)]。

例如，在 $M/M/1$ 队列的情况下，可以解析地证明，系统中稳态平均人数是：

$$L=\rho/(1-\rho)$$

我们在图 1.49 中画出了 L 对 ρ 函数图。注意，显然 L 不是 ρ 的线性函数，而且对于 $\rho>0.8$，L 的图形是呈指数增长的。虽然 L 的表达式是专为 $M/M/1$ 队列的，但从图 1.51 所看到的非线性特征一般说来表征了排队系统。

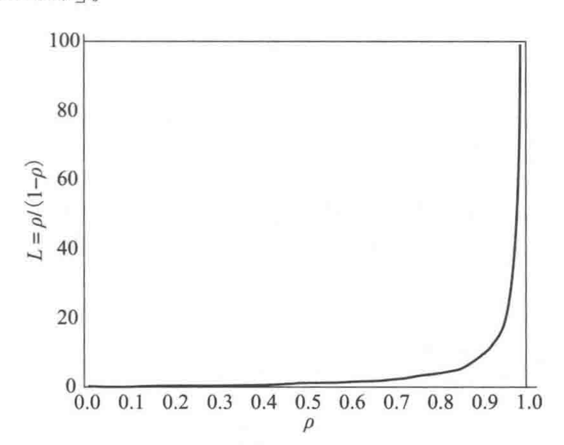


图 1.49　$M/M/1$ 队列的 $L=\rho/(1-\rho)$ 的图形

解析解的另一个有趣的(也是有启发性的)例子是，$M/G/1$ 队列稳态平均延时为：

$$d=\frac{\lambda\{\mathrm{var}(S)+[E(S)]^2\}}{2[1-\lambda E(S)]}$$

其中，$\mathrm{var}(S)$表示服务时间分布的方差[此式的推导参见，例如，Ross(2003，第 508 页)]。于是，我们可以看到，如果 $E(S)$很大，则拥塞(此处用 d 量度)就会更大；这当然是预期的结果。这个式子也带来一个也许不太明显的事实，即，即使平均服务时间保持不变，如果服务时间分布的可变性很大，拥塞程度也会增加。直觉上看，如果服务时间随机变量具有很高的可变性，就会有更大的机会取到大值(因为它必须是正数)，这意味着(单)服务台长时间阻塞，引起队列加长。

习题

1.1 请说明，按照图 1.1 中的可能性，为研究以下每个系统，你认为哪一个是最有效的方法，并讨论为什么？
　(a) 一个实际工厂的小部门；
　(b) 一个严重堵塞的高速公路立交桥；
　(c) 一个实际医院的急诊室；
　(d) 一个送比萨饼的店铺；
　(e) 机场汽车租赁公式的巴士；
　(f) 战场通信网络。

1.2 对于习题 1.1 中的每一个系统，假设决定通过仿真模型来研究。讨论仿真应当是静态的还是动态的，是确定性的还是随机的，以及是连续的还是离散的。

1.3 对于第 1.4 节中的单服务台排队系统，定义 $L(t)$是 t 时刻系统中顾客总数量(包括 t 时刻在队列中和正在接受服务的顾客，如果有的话)。
　(a) 请问 $L(t)=Q(t)+1$ 正确吗？为什么正确或不正确？
　(b) 考虑第 1.4.2 小节中的手工仿真的相同实现，画出时间 0 到 $T(6)$之间的 $L(t)$对 t 的图(类似于图 1.5 和图 1.6)。
　(c) 根据(b)中你画的图，计算$\hat{l}(6)$＝在时间区间[0，$T(6)$]系统中顾客的时间平均数。$\hat{l}(6)$的估计是什么？
　(d) 扩展图 1.7，以表明$\hat{l}(6)$在仿真中是如何计算出来的。

1.4 对于第 1.4 节的单服务台排队，假设我们不想估计队列中的期望平均延时；模型的结构和参数保持不变。这会改变状态变量吗？如果是，如何改变？

1.5 对于第 1.4 节的单服务台排队，令 W_i 为完成服务的第 i 个客户在系统中的总时间，包括该顾客在队时间加上接受服务时间。考虑第 1.4.2 小节中的手工仿真的相同实现，计算 $m=5$ 的 $\hat{w}(m)$($\hat{w}(m)$为离开系统的前 m 个顾客在系统中的平均时间)；通过适当地扩充图 1.7 来做。如果真的这样，这如何改变状态变量？

1.6 从图 1.5 可知，显然队列的最大长度为 3。写出此量的一般表达式(对 n 个延误的停止规则)，扩大图 1.7，使得在仿真过程中可以系统性地计算此表达式。

1.7 修改第 1.4.4 小节中的单服务台排队的代码，以便另外计算和输出下列性能的度量：
(a) 系统中时间平均顾客数(见题 1.3)；
(b) 系统中平均总时间(见题 1.5)；
(c) 最大队列长度(见题 1.6)；
(d) 队列中最大延误时间；
(e) 系统中最大时间；
(f) 队列中超过 1 分钟延误的顾客的比例。
运行该程序，使用附录 7A 中给出的随机数生成器。

1.8 第 1.4.3 小节中生成均值为 β 的指数随机变量算法返回值为 $-\beta\ln U$，其中，U 是随机变量 $U(0,1)$。这个算法对于改成返回值为 $-\beta\ln(1-U)$，仍然正确，为什么？

1.9 运行第 1.4.4 小节单服务台排队仿真 10 次，在大部分主程序外加上循环，只从初始化前开始，调用报告生成器后正好截至。讨论其结果。(这称为独立重复仿真 10 次)

1.10 对于第 1.4 节的单服务台排队仿真，假设设施早上 9 点开门(称此时间为 0)，下午 5 点关门，但继续运行，直到在下午 5 点还在的顾客(在服务中或在队列中)完成服务为止。修改代码以反映此停止策略，并估计与原来一样的性能度量。

1.11 对于第 1.4 节中讨论的单服务台排队仿真，假设排队房间只能容纳 2 个人，顾客到达看到队满就直接放弃(这称做“阻塞”)。仿真此系统，停止规则是正好 480 分钟，估计与第 1.4 节相同的量，以及被阻塞的顾客的期望数量。

1.12 考虑第 1.5 节的库存仿真。
(a) 对具有这些参数的本模型，在一个时刻始终不会有多于一个订单未处理(即以前的订购，但还没有交付)，为什么？
(b) 详细说明，如果交付延时是 0.5 到 6 个月之间均匀分布(而不是 0.5 到 1 个月)，那么必须做出什么样的改变；不考虑模型的其他改变。订货决策应当仅仅基于存储水平 $I(t)$ 吗？

1.13 修改第 1.5 节的库存仿真，使其可以对每个 (s,S) 策略进行 5 次重复运行；见题 1.9，讨论其结果。哪一种库存策略是最好的？你确定吗？

1.14 服务设施由两个串行的服务台组成(一前一后)，每一个都有自己的先入先出的队列(见图 1.50)。在服务台 1 完成服务的顾客将进入服务台 2，而在服务台 2 完成服务的顾客离开此设施。假设顾客到达服务台 1 的到达间隔时间是均值为 1 分钟的独立同分布的指数随机变量。顾客在服务台 1 的服务时间为均值为 0.7 的独立同分布的指数随机变量，而在服务台 2 的服务时间为均值为 0.9 的独立同分布的指数随机变量。运行此仿真严格 1 000 分钟，并估计每个服务台顾客在队列中的期望平均延误时间，队列中顾客的期望时间平均数，以及期望利用率。

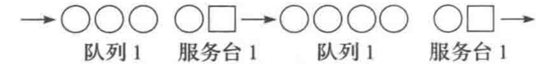


图 1.50 一前一后的排队系统

1.15 在习题 1.14 中，假设从服务台 1 出来到达队列 2(或者到服务台 2)有一个移动时间。假设该移动时间是 0 到 2 分钟之间的均匀分布。修改仿真，并在同样条件下重新运行，以得到同样的性能度量。那么所需要事件表的维数(即长度)是多少？

1.16 在习题 1.14 中，假设服务台 2 不设置等待队列。即如果顾客接受完服务台 1 的服务，并看到服务台 2 空闲，那么与以前一样，她会直接进入服务台 2。然而，服务台 2 忙于另一个顾客时，接受完服务台 1 的服务的顾客必须待在服务台 1，直到服务台 2 做完。这叫做封锁(blocking)。当一个顾客被封锁进入服务台 2 时，她不会接受服务台 1 任何更多的服务，但阻止服务台 1 为队列 1 中的第一个顾客服务，如果有的话。而且，在封锁期间，“新”顾客会继续到达队列 1。计算和习题 1.14 中同样的 6 个性能度量。

1.17 对于第 1.5 节的库存系统，假设如果库存水平 $I(t)$ 在月初小于 0，公司向供应商发出加急订单(如果 $0<I(t)<s$，公司仍会提交普通订单)。订购 Z 件的加急订单花费公司 $48+4Z$ 美元，但现在的交货延时是[0.25, 0.50]月的均匀分布。运行所有九种策略的仿真，并估计每月期望平均总成本，缺货即 $I(t)<0$ 的期望时间比例和加急订单的期望订货数。请问，加急订货值得吗？

1.18 对于第 1.5 节的库存系统，假设库存物品会过期的，其保质期是在 1.5 到 2.5 个月间的均匀分布。那么，如果一个物品的保质期为 l 个月，那么它在仓库放置 l 个月后，它就坏了，从而对公司就没

有价值了(注意，不同的货物会有不同的保质期)。公司只会在销售前检查货物才会发现它们是否坏了。如果货物确定坏了，那么它将被丢掉，再检查下一个库存的货物。假设库存中的货物按先入先出的方式处理。重复 9 次仿真运行并假定成本和从前一样。还要计算因为坏了而被丢掉的货物有多大比例从仓库清理出来。

1.19 考虑有 $s(s\geqslant 1)$个并行服务台的服务设施。假设顾客的到达间隔时间是均值为 $E(A)$的独立同分布的指数随机变量，而顾客的服务时间(不考虑服务台)是均值为 $E(S)$的独立同分布指数随机变量。如果顾客到达并发现一个服务台闲，顾客立即开始服务，如果有几个服务台可用，那么会选择最左边(编号最小)的空闲服务台。否则，客户会进入向所有服务台提供顾客的单一先入先出的队列尾部，等待接受服务(这称为 $M/M/s$ 队列；见附录 1B)。写一个通用程序来仿真这个系统，基于 n 个延误完成的停止规则，估计队列中的期望平均延误，队列中期望时间平均人数和每个服务台的期望利用率。量 s，$E(A)$，$E(S)$，和 n 应为输入参数。对 $s=5$，$E(A)=1$，$E(S)=4$，和 $n=1\,000$，运行此模型。

1.20 重复习题 1.19，但现在假设到达顾客发现有多于 1 个服务台空闲，他进行等概率选择。例如，如果 $s=5$，顾客到达发现 1，3，4 和 5 号服务台空闲，那么他将以 0.25 的概率选择这些服务台的每一个。

1.21 顾客户到达一个有三个并行出纳台的银行。

(a) 如果有单一先入先出的队列通向所有的出纳台，那么此系统的仿真模型所需要的事件表维数(即长度)是多少?

(b) 如果每一个出纳台有其自己的先入先出的队列，并且顾客可以从一个队列换到(跳队)另一个队列(参见第 2.6 节关于换队规则)，那么事件表需要的维数是多少？假设换队不花时间。

(c) 假设换队花 3 秒钟，重复(b)。

1.22 一个制造系统有 m 台机器，每一台都会随机地发生停机检修。在停机之前，机器运行时间的数量为均值 8 小时的指数变量。有 s(s 是固定的正值)名修理工来修理停工的机器，为了完成一台机器的修理，花一名修理工的时间是均值为 2 小时的指数变量；一台停工的机器委派的修理工不多于一名，即使有空闲的其他修理工。如果在某个时刻，停工检修的机器数量多于 s，这些机器形成“先入先出”的“修理”队列等着第一个可用的修理工。另外，无论系统中发生什么事情，正在修理停工机器的修理工一直工作，直到修好当前的机器为止。假设一台机器停工 1 小时花费系统 \$50，雇用一个修理工 1 小时需花 \$10(不管修理工是否实际工作，都要按小时支付工资)。假设 $m=5$，但在写通用代码时，通过改变输入参数，把 m 的值调高到 20。对于 $s=1, 2, \cdots, 5$ 每一种雇人策略，仿真此系统 800 小时，以确定哪种策略得到期望每小时成本最小。假设在 0 时刻，所有的机器恰好都刚刚维修完。

1.23 对于习题 1.10 的设备，假设服务员通常有 30 分钟午餐休息，第一次在中午 12 点后，此时设备空闲。然而，如果服务员在下午 1 点前还没有去吃午饭，那么服务员将在下午 1 点完成正在服务中的顾客后去吃饭(假设在这种情况下，下午 1 点所有在队列中的顾客都要等待，直到服务员返回)。如果顾客到达时服务员在吃午饭，则顾客可能立即离开而没有得到服务；这叫做阻塞。假设顾客是否被阻塞取决于服务员返回前剩余时间的数量(服务员张贴告示，他午饭后返回的时间)。特别是，一个在午饭期间到达的顾客会以如表 1.5 所示的概率被阻塞。

表 1.5

服务员返回前的剩余时间	顾客阻塞的概率
[20，30)	0.75
[10，20)	0.50
[0，10)	0.25

在第 1.5.2 小节中讨论的随机整数发生算法可以用于确定顾客是否阻塞。一个更简单的方法，参见第 8.4.1 小节。运行此仿真并同前面一样估计相同的性能度量(注意，在吃午饭时间服务员不是忙，那么计算队列中时间平均数所用数据要去掉午餐休息时间)。另外，估计被阻塞顾客的期望数。

1.24 对于第 1.4 节的单服务台排队设施，假设客户的服务时间在其到达时已知。完成顾客的服务后，服务台从队列中(如果有的话)选择最小服务时间的顾客。运行此仿真，直到 1 000 名顾客完成他

们的延误，并估计队列中的期望平均延时，队列中期望时间平均人数和在队列中延误超过 1 分钟的顾客数量期望比例值。(这个优先级排队策略称为最短作业优先)

1.25 对于习题 1.14 中串联的队列，假设在服务台 2 完成服务的顾客对他的整个服务以概率 0.2 有抱怨，且该顾客必须重新在两个服务台完整地服务(至少一次)。定义顾客在队列(在一个特殊的队列中)中的延误时间为所有经过服务设施的顾客的总延时。仿真该服务设施的下列每一种情况(估计与前面一样的性能度量)：

(a) 服务不满意的顾客进入队列 1 的队尾。

(b) 服务不满意的顾客进入队列 1 的队首。

1.26 一个服务设施包括两个 A 类服务台和一个 B 类服务台(在心理学意义上并不是必要的)。假设顾客到达设施的时间间隔是均值为 1 分钟的独立同分布指数随机变量。到达后，顾客确定为 1 类顾客或 2 类顾客，概率分别是 0.75 和 0.25。1 类顾客能被任意服务台服务，但如果 A 类服务台可用就选 A 类。1 类顾客的服务时间为均值为 0.8 分钟的独立同分布的指数随机变量，不管是哪类服务台。发现所有服务台都在忙的 1 类顾客进入类型 1 顾客的单一先入先出队列。2 类顾客需要 A 类服务台和 B 类服务台同时服务。2 类顾客的服务时间服从 0.5 到 0.7 之间的均匀分布。2 类顾客到达发现 A 类服务台忙或 B 类服务台忙时，则进入类型 2 顾客的单一先入先出队列。任何顾客完成服务后，如果是 2 类顾客且 A 类服务台与 B 类服务台均闲，则给予优先。否则，1 类顾客优先。仿真该设施正好 1 000 分钟，并估计队列中期望平均延误时间和每种类型的顾客在队列中的期望时间平均人数。还要估计每一个服务台花在每一类顾客上的时间的期望比例。

1.27 一个超市有两个付款台，普通的和快速的，每个付款台有一个结算员，如图 1.51 所示。普通顾客的到达间隔时间为均值 2.1 分钟的指数分布，服务时间为均值 2.0 分钟的指数分布。快速顾客的到达间隔时间为均值 1.1 分钟的指数分布，服务时间为均值 0.9 分钟的指数分布。两种顾客到达过程是相互独立的。普通顾客到达发现至少有一个付款台空闲，就立即开始服务；如果两个都空闲，就选择普通付款台；普通顾客到达发现两个付款台都忙，就加入普通队列尾。类似地，快速顾客到达发现一付款台空闲，则进入服务；如果两个付款台均为空闲，则选择快速付款台；快速顾客到达发现两个付款台都忙，则加入快速队列尾，即使这个队列比普通队列长。当任意一个结算员完成一个顾客服务，如果队列有人，他会从他的队列中取下一位顾客；如果他的队列空了，但另一队列未空，他取另一队列中的第一位顾客；如果两个队列都空，他变成闲。注意，顾客的平均服务时间由顾客类型决定，并不是由结算员是普通的还是加急的来决定。最开始，系统是空闲的，仿真运行正好 8 小时。计算每一个队列中的平均延误时间，每一个队列中时间平均人数，以及每一个结算员的利用率。为进一步研究或改进该系统，你会有什么建议？(在 1983 年 6 月 21 日，克利夫兰的 Plain Dealer 在一个题为“快速结算胜过食品低价”故事中报告说，“超市中的购物者认为快速柜台结算要比有吸引力的价格更重要。根据食品市场研究所的调查，…最大一组购物者，占 39%，回答“快速结算，”…而 28%的购物者说物好或低价”…(这反映)站在队中等着付钱引起愤怒)



图 1.51 超市付款台操作

1.28 一个单泵加油站全天服务并有两种顾客。每 30 分钟(精确)来一辆警车，第一辆警车在第 15 分钟到达。常规(非警车)车以均值为 5.6 分钟的指数分布间隔时间到达，第一辆常规车在时间 0 到达。对于所有的车，泵的服务时间为均值 4.8 分钟的指数随机变量。到达车辆发现泵空闲，则直接进入服务，且常规车到达发现泵忙就加入到单一队列的末尾。但是，警车到达发现泵忙则加入到队列的前头，队列中的任何常规车前面[如果已经有其他警车在队列前头，则假设正在到达警车也进到这些警车的前面(这可能发生么?)]最初系统是空闲的，仿真运行到正好 500 辆车(任何类型)完成其在队列中的延误。分别估计每一种车在队列中的期望平均延误时间，队列中期望时间平均车(所有类型)数和泵的期望利用率。

1.29 有关电话的模型是如下类型的模型。在两个大城市 A 和 B 之间，有固定的 n 条长途电话线路。每一条线路都可以双向通话(即，可以在 A 或 B 进行呼叫)但在一个时刻只能进行一个呼叫，如图 1.52 所示。如果一个在城市 A 或 B 的人想要给另一个城市打电话，并且有一条线是开通的(即空闲)，那么呼叫立

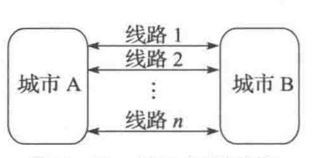


图 1.52 长途电话系统

即从这条线发出。如果 n 条线都忙，则此人会收到“请挂机，稍后再拨。”的录音提示；没有设备为下一条开通线排队，所以这些被封锁的呼叫者只能离开。从 A 到 B 的尝试呼叫之间的时间服从均值为 10 秒的指数随机分布，而从 B 到 A 的尝试呼叫之间的时间服从均值为 12 秒的指数随机分布。一次通话的长度服从均值为 3 分钟的指数随机分布，不考虑呼叫发出的城市。最开始所有线路都是开通的，仿真运行 12 小时；计算繁忙线路的时间平均数，忙的线路的时间平均比例，尝试呼叫(从两个城市)的总数，被封锁的呼叫个数，以及被封锁的呼叫的比例。请确定大约需要多少线路使得被封锁的尝试呼叫不多于 5%？

1.30　城市公交车到达维护设施的到达间隔时间是均值为 2 小时的指数分布。此设施包括一个检查站和两个相同的维修站，如图 1.53 所示。每辆公交车进行检查，检查时间均匀分布在 15 分钟到 1.05 小时之间；按单一先入先出队列进入检查站。历史上，检查时发现 30% 的公交车需要某种维修。两个并行的维修站由单一先入先出的队列进入，维修时间在 2.1 小时到 4.5 小时之间均匀分布。运行仿真 160 小时，并计算每个队列中的平均延误，每个队列的平均长度，检查站的利用率，以及维修站的利用率(定义为维修站处于忙碌状态的时间平均数的一半)。重复仿真 5 次。假设公交车的到达速率变为原值的 4 倍，即平均到达时间间隔降为 30 分钟，该设施还能应付吗？你不仿真可否回答这个问题？

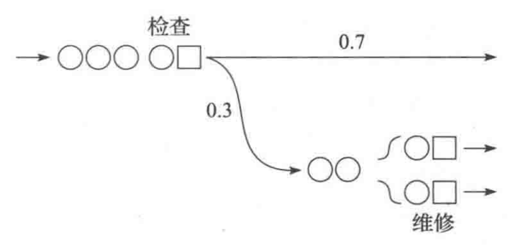


图 1.53　公交车维护站

第2章 复杂系统建模

2.1 引言

在第1章中我们一般性地介绍了仿真建模的问题，接着对两个特定的系统进行了建模和编码。这些系统非常简单，可以用通用语言直接编程，而不使用任何仿真软件或支持程序(随机数发生器除外)。然而大多数实际系统相当复杂，不用支持软件直接编程是一件十分困难而耗时的工作。

本章首先讨论一个在大多数仿真中都出现的活动：表处理，然后介绍一组ANSI标准C语言支持Simlib仿真语言，它负责管理某些标准的表处理任务，以及其他一些常见的仿真活动，例如处理事件表、累积统计、产生随机数和从不多的几个分布产生观测值，以及计算和输出结果。然后我们将Simlib仿真语言用于四个仿真例子，其中第一个例子就是第1.4节中的单服务台排队系统(包括它的原因是用熟悉的模型来说明Simlib仿真语言的使用方法)；后面三个例子则稍微复杂一些。

本章目的旨在说明如何能对更复杂的系统进行建模，以及表处理和Simlib实用函数如何辅助它们编程。我们使用像Simlib这样的软件包的意图仅仅是为了教学，它使读者可以快速进入更复杂系统的建模，并了解实际的仿真软件包是如何处理表和其他数据的。我们并不是认为Simlib像成熟的商业仿真软件(在第3章和第13章中讨论)那样综合或有效，或者可以用某种方式进行比较；事实上，Simlib的完整源代码在附录2A及图7.5中给出，也可从www.mhhe.com/law上下载(包括本章所有例子的代码)。也可从该网址下载本章所讨论的四个例子的FORTRAN 77版本的Simlib及其相关文档说明和代码。

2.2 仿真中的表处理

第1章所考虑的仿真模型的确相当简单，模型包括的除事件表外要么是一个记录表要么没有表。此外，这些表中的记录只包含一个属性，且都按照先进先出(FIFO)方式处理。在排队例子中，有一个FIFO表，它包含队列中所有等待顾客的记录，且每个顾客的记录只包含一个属性：到达时间。在库存例子中，除了事件表就没有其他表了。然而，大多数复杂仿真需要多个表，每个表可能包含众多记录，进而每条记录有可能包含多个属性。此外，表处理通常需要非FIFO的方式。例如，在一些模型中，需要能删除某一特定属性的最小值的记录(除了记录存放于表中的时间以外)。这样数量庞大的信息如果不能有效地存储和操作，模型将非常耗费时间和存储空间，以至于仿真研究变得不可行。

在第2.2.1小节中我们将讨论在计算机中存储记录表的两种方法：顺序分配及链式分配，并解释为什么后者更利于复杂仿真。在第2.2.2小节中我们给出一个链式存储分配的方法，其对开发简单的基于C的仿真“语言”Simlib来说是足够了(见第2.3节)。这种语言，用短短几个小时的学习就可完全掌握，提供了对第3章和第13章中讨论的专用仿真软件的本质的洞悉，学习专用仿真软件则需要多得多的时间。更重要的是，Simlib为我们提供了一个工具，以解释如何对一个比第1章讨论的系统更复杂的系统进行仿真。

2.2.1 计算机中存储表的方法

计算机中存储记录表的基本方法有两种。第1章中用到的为*顺序分配法*，表中的记录存入物理相邻的存储位置上，一个接一个。这是第1.4节中用到的方法，用于队列中顾客

的到达时间表。

在链式分配法中，表中的每个记录除包含其本身通常的属性之外，还有一个指针(或链接)，给出了该记录与表中其他记录的逻辑关系。表中逻辑上彼此相接的记录并不需要存储在物理上相邻的位置。链式分配的进一步讨论将在第 2.2.2 小节中给出。表的链式分配对仿真建模有如下优点。

- 处理某一类表所需的时间可大大缩短。例如第 1.4 节中的排队例子，每当一个顾客完成服务(且留下一个非空的队列)时，我们必须把到达时间数组中的每一项向上移动一个存储位置；如果队列很长的话，数组可能包含大量的记录，这是非常低效的做法。正如在例 2.1 中我们将会看到，链式分配使这类数组的处理得到加速。
- 对于事件表同时包含大量事件记录的仿真模型，我们可以大大加速事件表处理；进一步讨论可参见例 2.2 以及第 2.8 节。
- 对于某些仿真模型，存储所需的计算机内存将大大减少；参见第 2.2.2 小节末尾的讨论。
- 链式分配提供了一个通用框架，使得同时存储、操作多个表易行，尽管不同表中的记录可能用不同方式处理。这种通用性也是所有主流仿真软件都采用链式分配法的原因之一。

2.2.2 链式存储分配

在本节，我们对链式存储分配进行讨论，它足以满足下节中讨论的简单的基于 C 的仿真“语言”Simlib 的开发。对表处理原理的更完整更一般的讨论，请参见，例如，Knuth (1997，第 2 章)。

假设记录表以数组形式存储，数组的行对应记录，列对应组成记录的属性(或数据域)。对第 1.4 节中的排队仿真，队列中等待的每个顾客在到达时间表中有一条记录，每条记录由单一属性组成，相应于顾客的到达时间。通常，一个顾客的记录可能还有其他属性，例如年龄、优先级、服务要求等。

一个记录表，如果每条记录都有一个与它相关的前链和后链，则称为*双链的*。一个特定记录的*后链*(或称*前向指针*)给出了在该记录的数组中的物理行号，该记录逻辑上后接给定记录。如果没有后接给定的记录，则后链设为零。一个特定记录的*前链*(或称*后向指针*)给出了在该记录的数组中的物理行号，该记录逻辑上前接一个给定的记录。如果没有前接给定的记录，则前链设为零。表中逻辑第一的记录，其数组中的物理行号用*头指针*标识，当表没有记录时，头指针设为零。表中逻辑上最后的记录，其物理行号用尾指针来标识，当表为空时，尾指针设为零。

在任意给定时刻，一个表可能只占用物理存储的数组行的一个子集。数组的“空”行链接在一起组成*可用空间表*(LAS)，以备将来使用。LAS 通常按照 LIFO(后进先出)方式处理：这就意味着当一行需要存储一条新加记录时，该行从 LAS 的头部取出；当一行不再需要存储记录时，该行重返 LAS 的头部。由于所有操作都是在 LAS 头部进行，因此既不需要尾指针也不需要前链(我们称这样的表是*单链表*)。在仿真的时刻 0，数组的所有行都是 LAS 的成员，行 i 的后链设为 $i+1$(最后一行例外，它设为 0)，所有前链都设为 0，LAS 的头设为 1(只有当某特定行被一条记录占用时，该行的前链才设为一个正整数)。在支持动态存储分配的语言(如 C 语言)中，LAS 可以将所有内存视为可用于动态分配的。

例 2.1 对于第 1.4 节的排队仿真，考虑队列中等待服务的所有顾客的表。该表的每条记录有单一属性“到达时间”。假设在仿真的 25 时刻，队列中有 3 名顾客，到达时间分别为 10、15 和 25，这些记录物理上分别存于有 5 行 1 列的数组的第 2 行、第 3 行和第 1 行(为使图 2.1 易于处理，我们假设在任意时刻队列中都不会超过 5 个顾客)。第 4 行和第 5 行是 LAS 的成员，情况如图 2.1 所示。注意，该表的头指针等于 2，第 2 行记录的后链等于 3，第 3 行记录的前链等于 2，以此类推。

假设仿真的下一事件(时刻 25 之后)是一个顾客在时刻 40 到达，且我们希望向该表添加一个适当的记录，该记录用 FIFO 方式处理。由于 LAS 的头指针等于 4，该到达顾客的记录将被置于数组的物理第 4 行，且 LAS 的头指针现在置为 5，它是第 4 行后链的值。由于新记录添加到该表的尾部，该表的尾指针现在等于 1，第 1 行记录的后链设为 4，新记录(即(物理)第 4 行的记录)的前链设为 1，新记录的后链设为 0，且该表的尾指针设为 4。做了这些变化之后的两个表的状态如图 2.2 所示。

假设仿真的下一事件(时刻 40 之后)为已经开始服务的顾客在时刻 50 完成服务(至少从时刻 25 之后开始的)，因而我们要删除表头的顾客记录以使该顾客开始服务。由于该表的头指针等于 2，且(物理)第 2 行记录的后链等于 3，第 2 行记录的到达时间用于计算即将进入服务的顾客的延误时间(该延误时间是 50 减 10)，该表头指针设为 3，第 3 行记录的前链设为 0，第 2 行(不再需要)被置于 LAS 的头部，即将它的头指针设为 2 且第 2 行的后链设为 5(LAS 之前的头部)。做了这些变化之后的两个表的状态如图 2.3 所示。

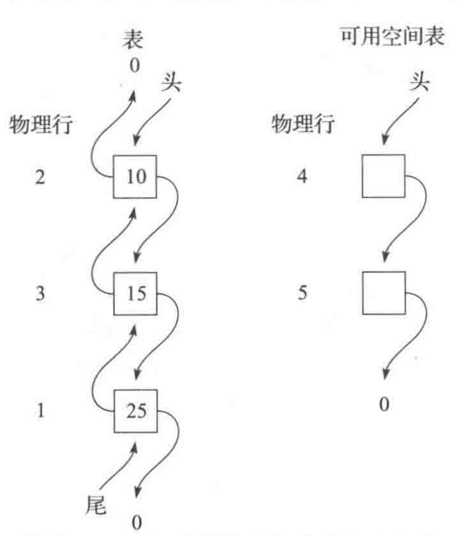


图 2.1　在 25 时刻排队仿真的表的状态

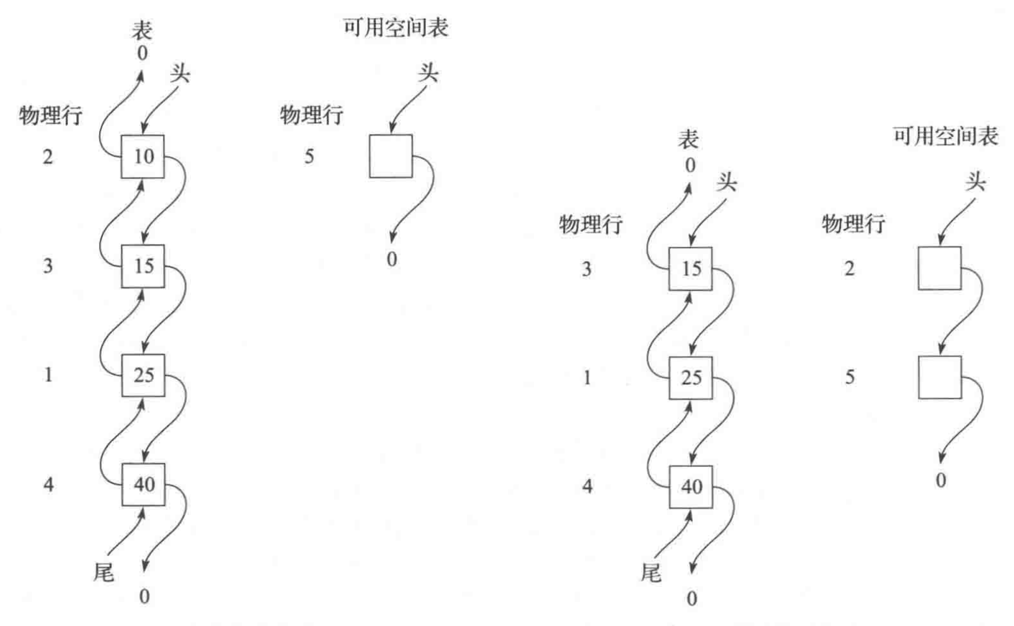


图 2.2　在 40 时刻排队仿真的表的状态

图 2.3　在 50 时刻排队仿真的表的状态

因此，删除表头的一个记录总是只需要设置 4 个链接或指针。将此方法与第 1 章中将(顺序)表中的每个记录向上移 1 位的笨办法作一下对比，譬如说，如果该表包含 100 个记录，则它比链表法耗时多得多。◀

尽管将队列如在上例中那样以链表存储似乎相当自然，下例将说明事件表如何也可像链表那样处理。

例 2.2 对第 1.5 节中的库存仿真，事件表存放在一个数组中，四种事件类型的每一种都分配一个物理地址。如果一个事件当前还未安排发生，则其进入表的时间设为 $+\infty$(在计算机中用 10^{30} 表示)。然而，对很多用通用语言编写的复杂仿真，以及第 3 章和第 13 章中使

用专用仿真软件的仿真，事件表以链表形式存储，并以事件时间的递增顺序排列。那么，事件时间为$+\infty$的事件显然不在事件表中。此外，由于事件表按照时间(属性 1)的递增顺序排列，下一个发生的事件将总在队首，因此我们只需删除这个记录来确定下一事件的时间(属性 1)及其类型(属性 2)。例如，设库存仿真的事件表存储于一个 4 行 2 列的数组中，列 1 存储属性“事件时间”，列 2 存储属性“事件类型”，即 1、2、3 或 4。假设在 0 时刻我们知道第一个产品需求(事件类型 2)发生在 0.25 时刻，第一次库存评估(事件类型 4)将在 0 时刻立即发生，仿真将在 120 时刻结束(事件类型 3)，没有尚未交货的订单安排到达(事件类型 1)。在 0 时刻初始化之后事件表和 LAS 的状态如图 2.4 所示。注意事件类型 1 并不包含在事件表中，事件类型 2 在数组的(物理)第 1 行，因为它是放在事件表中的第一个事件记录。

要确定在时刻 0 发生的下一(第一个)事件，从事件表中删除第一条记录，仿真钟更新到该条记录的第一个属性，即仿真钟设为 0，下一个发生的事件的事件类型设为该记录的第二个属性，即设为 4，包含这条记录的第 3 行置于 LAS 头部。由于下一事件的类型为 4，库存评估时间即将发生(在时刻 0)。假设 0 时刻发出一个订货，则将于 0.6 时刻由供应商送达。将该订单到达事件置于事件列表中，首先 0.6 和 1 分别置于第 3 行(LAS 的头部)的第 1 列和第 2 列。然后按向下推进事件表逻辑(用后链)，直到找到正确的位置时，将这一新记录加到事件表中。特别是，首先将新记录的属性 1(0.6)与第 1 行记录的属性 1(0.25)作对比。由于 0.6>0.25，新记录将置于事件表中更靠下的位置。接下来，0.6 与第 2 行记录的属性 1(120)作比较(注意第 1 行记录的后链为 2)。由于 0.6<120，通过调整这三个记录的前后链，新记录在逻辑上将置于物理第 1 行和第 2 行记录中间。完成这些之后，另一个库存评估事件安排在 1 时刻，并用类似于订单到达事件的方式将其置于事件表中。在 0 时刻所有操作完成之后的两个表的状态如图 2.5 所示。

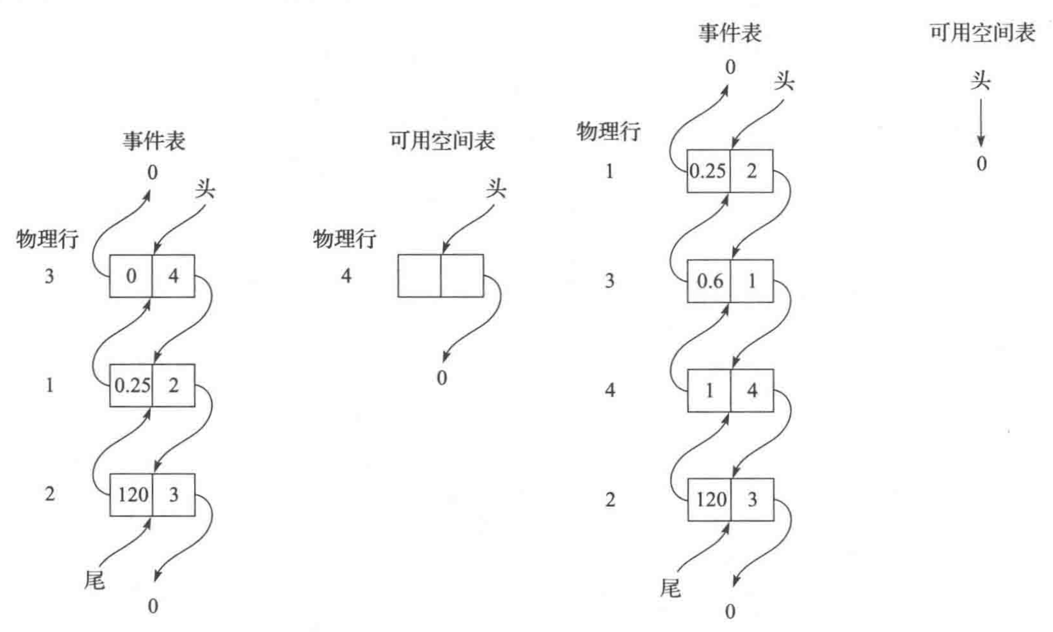


图 2.4　库存仿真，在 0 时刻初始化后，表的状态

图 2.5　库存仿真，在 0 时刻做完所有处理后，表的状态

在上述讨论中，单一表用数组存储，其空行是 LAS 的成员，但是我们幸好可以在同一个物理数组中同时存储很多不同列表。只有单一的 LAS，每个表的首尾用不同的头指针和尾指针标识。这种方法使得某些应用的存储空间大大节省。例如，假设一个仿真需要 20 个表，每个表包含 100 条记录，每条记录有 10 个属性。用顺序存储法(如第 1 章中那样)，会需要 20 个 100 行 10 列的数组，则总存储需求为 20 000 个位置。然而，假设在任意给定

时刻，所有可用行平均只有25%被实际使用。那么，用另一种方法是，可以将这20个表存储在一个由1 000行10列组成的数组中。这种方法需要10 000个存储位置用于数组，外加2 040个位置用于链和指针，总存储量为12 040个位置。进一步的，在仿真的一个特定时刻，某些表可以占用的行数超过其“平均”分配数，而不会导致内存溢出。

下节中讨论的简单仿真语言Simlib以动态内存方式存储所有表(包括事件表)，当一个新记录需要在表中存储时则分配，当记录从表中删除后则释放。

2.3 简单仿真语言：Simlib

本节我们将介绍一个易于理解的基于C语言的仿真“语言”Simlib，它实现了第2.2.2小节中所述的链式存储分配概念。该语言使得易于进行表中的记录排序(记录可以排在表的前头，表的最后，或者使表按照规定属性递增或递减的顺序进行排列)、从表中删除一条记录(无论是表第一条记录，还是表的最后一个记录均可删除)、处理事件表、计算有关变量的离散时间统计(如在排队系统中队列中的平均延误时间)、计算有关变量的连续时间统计(如在库存系统中货物的时间平均数)、产生本章和第1章例子中用到的随机变量，以及需要时提供一份“标准”格式的输出。尽管Simlib提供了很多专用仿真软件(参见第3章和第13章)的重要特性，它的设计既没有考虑完整性，也没有考虑计算效率。我们在此介绍它的原因是为了比较深入地了解仿真软件的操作，并提供一个工具，以便理解如何对比第1章中的系统复杂得多的系统进行仿真。

Simlib的核心是一系列双链表，在动态内存中它们驻留在一起，当新记录进入该表排序中时分配空间，当记录从该表中删除时释放空间。最多有25张表，每张表中的记录最多可以有10个属性，所有数据都以浮点类型存储。由于Simlib使用动态存储分配，所有表记录的总数仅由计算机可用内存总量限制。

表25总是保留用于事件表，属性1为事件时间，属性2为事件类型。进一步，该表按照属性1(事件时间)递增的顺序存储，使得最上面的(第一个)记录总是用于查阅下一事件。

要使用Simlib，用户必须引用(# included) simlib. h文件。该文件依次引用(# included)simlibdefs. h文件。这些文件在附录2A的图2.47和图2.48中给出，它们包含了(除了其他方面以外)表2.1所示的与用户关系密切的变量和常量的声明及定义。

表 2.1

sim_time	仿真钟，浮点型变量，由Simlib定时函数更新(见后面定时函数的讨论)
next_event_type	下一事件的类型，整数型变量，由Simlib定时函数确定(见后面定时函数的讨论)
transfer[i]	浮点数组，下标为i=1，2，…，10(下标i=0不使用)，用于仿真中表的传进与传出，其中transfer[i]指的是记录的属性i。transfer数组在Simlib中还可用作地址以便将某类求和统计传送给用户
maxatr	仿真模型的任意表中的任意记录的属性最大值，整数型变量。如果用户在主函数中未设置maxatr，则使用默认值为10；maxatr不能超过10。如果可能的话，设置maxatr的值小于10，将使程序运行更快。由于Simlib编程方式的原因，用户规定maxatr必须至少为4
list_size[list]	当前“list”表中的记录数，整数型数组，由Simlib自动更新。用户应该不需要改变list_size[list]的值，适当情况下只能查询其值。例如，如果表3设为表示一个队列，则list_size[3]将一直表示该队列中的数目，使得我们可以通过检查list_size[3]是否为0来判断队列是否为空
list_rank[list]	属性值，如果有的话，是由Simlib的函数list_life()根据该属性值对表“list”中的记录进行排序(按递增或递减的顺序)；这是个整数型数组。例如，如果表4使记录按照属性2排列，则在使用list_file()在list 4中存储记录之前，用户需要设置list_rank[4]等于2；在list_file()函数中的一个哑元控制是递增排序还是递减排序，典型地，list_rank[list]由用户在主函数中设置。如果表“list”没有排序(即记录只在头或尾插入)，则list_rank[list]根本不需要设置。注意list_rank[25]在Simlib初始化例程init_simlib里设为1，因为事件表(表号25)的属性1总是事件时间，且我们希望使事件表保持按照事件时间的递增顺序来排序
FIRST	符号常量，用于在表头加入或删除一条记录的选项，在simlibdefs. h中自动设为1

（续）

LAST	符号常量，用于在表尾加入或删除一条记录的选项，在 simlibdefs. h 中自动设为 2
INCREASING	符号常量，用于根据 list_rank 数组中的规定的属性以递增顺序进行表排序的选项，在 simlibdefs. h 中自动设为 3
DECREASING	符号常量，用于根据 list_rank 数组中的规定的属性以递减顺序进行表排序的选项，在 simlibdefs. h 中自动设为 4
LIST_EVENT	符号常量，用于事件表 25 的编号，在 simlibdefs. h 中自动设为 25
EVENT_TIME	符号常量，用于表示事件表中事件时间的属性号，在 simlibdefs. h 中自动设为 1
EVENT_TYPE	符号常量，用于表示事件表中事件类型的属性号，在 simlibdefs. h 中自动设为 2

这些变量和数组在仿真时由用户适当地使用或设置，且必须如在 simlib. h 中声明的那样使用上面给出的名称及类型。

共有 19 个函数组成了 Simlib，每个函数设计成执行一个频繁发生的仿真活动。

- init_simlib　在每次仿真运行开始时，这个函数由用户所写的主程序调用，为表分配存储空间，对于每个表初始化前后链以及头尾指针，初始化仿真钟为 0，设置 list_rank[EVENT_LIST]为 EVENT_TIME 以保证事件列表按照事件时间排序，并设 maxatr 为默认值 10。此外，函数 sampst 和 timest(见下面的讨论)的统计累加器均设为 0。
- list_file(option，list)　这个函数将一条记录的属性放到由哑元“option”控制的位置上，该记录在 transfer 数组中，用户要将其放到表“list”中。亦即，当 list_file 被调用时，transfer[i]被视作新记录的属性 i 填入表“list”中，i＝1，2，…，maxatr。表 2.2 所示的为可用的选项。

表　2.2

option	活动
1(或 FIRST)	将 transfer 数组中的记录存于“list”表中当前第一条记录之前
2(或 LAST)	将 transfer 数组中的记录存于“list”表中当前最后一条记录之后
3(或 INCREASING)	将 transfer 数组中的记录存于“list”表，使得该表按 list_rank[list]属性的递增顺序排序，其中 list_rank[list]必须事先赋值(如果两条记录的 list_rank[list]属性相同，则按照 FIFO 规则存储)
4(或 DECREASING)	将 transfer 数组中的记录存于“list”表，使得该表按照 list_rank[list]属性的递减顺序排序，其中 list_rank[list]必须事先赋值(如果两条记录的 list_rank[list]属性相同，则按照 FIFO 规则存储)

因此，list_file(1，3)将把 transfer[1]，…，transfer[maxatr]存入表 3，使之成为表 3 的第一条记录；list_file(FIRST，3)与之相同。如果我们想要表 2 按照记录的属性 4 进行递减顺序的排序，则需要执行 list_file(DECREASING，2)，确保我们之前，比如在主函数中已将 list_rank[2]设为 4。最后，我们可以通过执行 list_file(INCREASING，LIST_EVENT)将一个事件放入事件表中，这之前需先将(至少)transfer[1]设为该事件的发生时间，并将 transfer[2]设为该事件的类型(但是，有一种更简便的方法将事件放入事件表中，参见下面 simlib 的 event_schedule 函数的描述)。表“list”中的新记录的存储是由 list_file 动态分配的。

- list_remove(option，list)　调用这个函数从“list”(一个整数)表中删除一个记录，并将其(属性)复制到 transfer 数组中。这个整型哑元“option”决定哪个记录将被删除，如表 2.3 所示。

表　2.3

option	活动
1(或 FIRST)	从“list”表中删除头条记录并置之于 transfer 数组
2(或 LAST)	从“list”表中删除末条记录并置之于 transfer 数组

在 list_remove 被调用后，transfer 中的元素，现在等于那些刚刚被删除的记录中的元素，典型地，它们将用于仿真中某些目的。例如，list_remove(2，1)删除了表 1 中的最后一条记录，并将其放入 transfer 数组中；list_remove(LAST，1)与之相同。由 list_remove 释放的存储空间是动态的，因为“list”表中刚被删除的记录所占的空间将不再需要。

- timing 这个函数由用户从主函数调用，完成第 1 章定时函数所做的事情，但现在定时函数是内部的 Simlib 函数，且它使用的事件表的结构与例 2.2 完全一致。确定下一事件的类型 next_event_type(一个 Simlib 整数型变量，在 simlib.h 中声明)，且更新仿真钟 sim_time 为该下一事件的时间。事实上，定时函数只是简单地调用 list_remove(FIRST，LIST_EVENT)以从表 25(事件表)中删除第一条记录；由于事件表是按照事件时间递增顺序排列的，我们知道它将是下一个即将发生的事件。因此，第一个事件记录的属性 1 到属性 maxatr 都放到 transfer 数组中，如果需要就能使用。特别地，利用除属性 1 和 2 以外的事件记录的其他属性有时是非常有利的；参见 2.6 节和 2.7 节，以及习题 2.3。
- event_schedule(time_of_event，type_of_event) 用户可以调用这个函数将一个类型为 type_of_event(整数型)安排在时间为 time_of_event(浮点型)发生的事件到事件表中。通常，调用 event_schedule 以安排一个仿真的将来事件，因此 time_of_event 具有 sim_time+time_interval 的形式，其中 time_interval 是从当前时间到事件发生时间的差值。如果事件表中用到事件记录的其他属性(非属性 1 事件时间，亦非属性 2 事件类型)，那么用户负责在调用 event_schedule 之前，将它们的值放入 transfer 中合适的位置当中。
- event_cancel(event_type) 该函数可用来取消(删除)具有类型为 event_type(整数型)事件的事件表中第一个事件，如果事件表中有这样的事件的话，将被取消的事件记录的属性放入 transfer 数组中。如果事件表中没有类型为 event_type 的事件，则函数 event_cancel 无动作发生。如果 event_cancel 找到一个类型为 event_type 的事件并将其取消，则函数返回一个整型值 1；如果未发现这类事件，则函数返回整型值 0。
- sampst(value，variable) 该函数收集并累加离散时间数据，例如队列中顾客延误时间。最多提供 20 个“sampst 变量”，并由整型哑元“variable”来维护、分别求和，以及索引。例如，一个模型可能包括三个独立的队列，sampst 变量 1，2 和 3 可用来分别收集并累加这三个队列中顾客的延误时间。sampst 的三类典型用途如下。

  在仿真过程中 每当观测到 sampst 变量“variable”的一个新值时(如队列中一次延误的结束)，其值放到浮点型哑元“value”中，sampst 被调用。例如，如果我们定义 sampst 变量 2 为模型中队列 2 的延误时间，那么将希望的延误时间放到浮点型变量 delay2 之后，我们可执行 sampst(delay2，2)。函数 sampst 在内部维护每个变量的独立寄存器，使得统计结果被累加以产生后面描述的输出。

  在仿真结束时 用户可以用想要的变量“variable”的负数来调用 sampst，以产生求和统计，按表 2.4 所示的方式放到 transfer 数组中。

表 2.4

i	transfer[i]
1	观测到的变量“variable”的均值(平均)
2	观测到的变量“variable”值的个数
3	观测到的变量“variable”的最大值
4	观测到的变量“variable”的最小值

此外，sampst 返回一个浮点型的均值(用其名字的方式)，为方便使用，将其放入 transfer[1]中，因为往往需要均值。例如，执行 sampst(0.0，－2)将把 sampst 变量 2 的求和统计放到 transfer 数组中，如上所述，返回均值；所要的求和统计，代表性的做法是由用户将其输出或者用作其他用途。注意这样使用 sampst 时，“value”的值被忽略了(技术提示：如果没有观测到变量“variable”的值，则其均值、最大值及最小值未定义。在这种情况下，sampst 返回均值为 0、最大值为 -10^{30}、最小值为 10^{30}。——译者注：此段文字疑有误，应为“最大值为 10^{30}、最小值 -10^{30}”)。

重置所有 sampst 变量累计器 当在仿真开始时，通过执行 sampst(0.0，0)，所有 sampst 变量的累计器重新初始化。注意，这是在时刻 0 时在 init_simlib 中完成的。这个功能在仿真过程中是非常有用的，如果我们只想在仿真已经“预热”一段时间之后再观察数据的话，正如第 9.5.1 小节所描述的那样，参见习题 2.7。

- **timest(value，variable)** 该函数类似于 sampst，但是它操作的对象代之以连续时间数据，诸如队列中顾客数，参见第 1.4.1 小节。同样，整型哑元“variable”最多有 20 个的“timest 变量”，需要时收集并累加这些“timest”变量的数据。例如，timest 变量 1、2 和 3 可以分别为队列 1、2 和 3 中的顾客数。与 sampst 同样，timest 有如下三个“典型的”用途。

在仿真过程中 每当 timest 变量“variable”获得一个新值时，我们必须执行 timest(value，variable)，其中浮点型哑元“value”包含该变量的新值(即改变之后的值)。例如，如果队列 2 的长度发生变化，浮点型变量 q2 作为一个到达或离去后的结果，我们应执行 timest(q2，2)以完成适当的累加。timest 的累加器在 init_simlib 中被初始化，并假设由 timest 跟踪的所有连续时间函数都初始化为 0。这可以改写，只要在调用 init_simlib 之后执行 timest(value，variable)就行了，其中 value 包含了所要求的(非零)timest 变量“variable”初始值(对每个所要求的 timest 变量均做此处理)。

在仿真结束时 用户可以用所要求的变量“variable”的负数来调用 timest，以生成求和统计，按表 2.5 所示的方式放到 transfer 数组中。

表 2.5

i	transfer[*i*]
1	所观测到的变量“variable”值的时间平均，更新到本次调用时
2	到本次调用时所观测到的变量“variable”的最大值
3	到本次调用时所观测到的变量“variable”的最小值

此外，timest 返回一个浮点型的时间平均(用其名字的方式)，为方便使用，将其放入 transfer[1]中，因为往往需要均值。例如，执行 timest(0.0，－2)，将把 timest 变量 2 的求和统计放到 transfer 数组中，如上面所说的那样，并返回时间均值；所要的求和统计，代表性的做法是由用户将其输出，或者用作其他用途。

重置所有 timest 变量累计器 当在仿真开始时，通过执行 timest(0.0，0)，所有 timest 变量的累加器重新初始化为 0。这是在时刻 0 时在 init_simlib 中完成的。注意，这里假设所有 timest 变量的值都为零，这可以改写，通过立即执行 timest(value，variable)重置 timest 变量 variable 的值为 value。

- **filest(list)** 该函数的典型用法是只在仿真运行结束时调用。它提供“list“表中记录条数的求和数据，并按照类似于 timest 的方式将其放到 transfer 数组中，如表 2.6所示：

表 2.6

i	transfer[i]
1	"list"表中的记录条数的时间平均值，更新到本次调用时
2	到本次调用为止"list"表中的记录条数的最大值
3	到本次调用为止"list"表中的记录条数的最小值

此外，filest 返回一个浮点型的时间平均值(以其名字的方式)，为了使用方便，将其放到 transfer[1]中，因为往往需要均值。从 Simlib 内部来说，它把表中的记录条数作为一个连续时间函数来对待，其值只可能在事件时间时刻增加或减少，这样才能理解表中记录条数的时间平均值这样一种说法，等等。此外，Simlib 以这种方式自动跟踪每张表，并在 filest 被调用时生成这些统计结果。谁关心队列长度的历史记录呢？这个功能带来相当多的方便，因为表中的记录条数往往有某些物理含义。例如，队列在仿真中通常用表来表示，那么在该表中的记录条数就等于队列中的顾客数；因此队列中顾客时间平均数和最大数就是表中记录条数的时间平均值和最大值。另一个例子是用于表示服务台的表，其中，当服务台忙时，表中有一条记录，当服务台闲时，表为空；因此，服务台的利用率就是该表中记录条数的时间平均值，因为其长度只可能为 0 或 1。这样一来，我们往往(并不总是)可以避免通过 timest 详实地跟踪一个连续时间函数。然而，如果相应的表只是为了统计收集的方便而建立的，尤其是当被跟踪函数加减的增量不是 1(如 1.5 节中模型的库存水平)时，由于添加和删除许多虚拟记录需要附加开销，恐怕应该用 timest 来替代 filest。另外，当被跟踪函数可以取非整型值时，必须用 timest 代替 filest(在 Simlib 内部，它将"list"表中的记录条数作为 timest 变量 20+list 来处理，因此实际上有 45 个 timest 变量，但是只有前 20 个对用户是可存取的。而 filest 仅是简单用变量=−(20+list)调用 timest 以获得"list"表中的统计信息)。

- out_sampst(unit，lowvar，highvar)　如果需要，可调用该函数以产生 sampst 变量 lowvar 到 highvar(含)的求和统计，并将其写入文件"unit"；lowvar 和 highvar 均为整数型参数。它生成"标准"输出格式(每行 80 个字符)，并免除最终调用 sampst 的需要(但并不是在仿真过程中调用)，还可以免除 fprintf 声明、格式化等的需要。使用 out_sampst 的一个弊端是输出的注释和布局不能控制或个性化。例如，out_sampst(outfile，1，3)将 sampst 变量 1、2 和 3 的求和统计写入 outfile 文件；out_sampst(outfile，4，4)用于写 sampst 变量 4 的求和统计。在本章后面的 simlib 仿真中，我们将给出使用(及避免)这种标准输出格式的例子。
- out_timest(unit，lowvar，highvar)　类似于 out_sampst，这个可选的函数可用于产生 timest 变量 lowvar 到 highvar 的文件"unit"的标准格式输出。
- out_filest(unit，lowfile，highfile)　该函数用 filest 来生成文件 lowfile 到 highfile 中的记录条数的求和统计，并将其存入文件"unit"中；lowfile 和 highfile 均为整数型参数。
- expon(mean，stream)　该函数返回一个来自均值为"mean"(浮点型参数)的指数型分布的浮点型观测值。整型参数 stream 为用户规定的随机数流的数目，第 7.1 节与附录 7A 中将更充分地进行讨论。目前，我们可将流的数目视作一个各自独立的随机数发生器(或随机数表)，以用于从指数型分布中生成期望的观测值。一般说来，对仿真中的一个特殊随机源"指定"一个随机数流是一个好主意，例如：stream1 用于到达间隔时间，stream2 用于服务时间等等，从而便于使用方差减少技术(参见第 11 章)。这些技术往往能在仿真的统计精度方面提供很大改进。此外，使用指定的流可以帮助程序校验(调试)。除了流的规范说明外，该函数与第 1.4.4 小节中由指数分布生成观测值的函数是一样的。Simlib 中有 100 个不同的流可用；即

"stream" 必须为 1～100(含)之间的一个整数，每个流的长度为 100 000 个随机数。

- random_integer(prob_distrib[]，stream)　该函数返回一个来自离散概率分布的整型观测值，该离散分布具有累计分布函数值，由用户在浮点型数组 prob_distrib 中定义。对于一个介于 1 和 25 之间的正整数 i，在调用 random_integer 之前，prob_distrib[i]应该由用户定义为生成一个小于或等于 i 的值的所要求概率。如果所要求的随机整数的范围为 1，2，…，k，其中 k<25，则 prob_distrib[k]应设为 1.0，且对于所有 j>k 不必再定义 prob_distrib[j]。整型参数 stream 介于 1～100 之间，给出所使用的随机数流。除了随机数流的说明之外，该函数和第 1.5.3 小节库存模型中用到的函数一样。
- uniform(a，b，stream)　该函数返回一个 a 与 b 之间(均为浮点型参数)的(连续)均匀分布的浮点型观测值。如前面一样，stream 是一个介于 1 和 100 之间的整数，给出了所用的随机数流。
- erlang(m，mean，stream)　该函数返回一个均值为 mean 的 m-厄兰分布的浮点型观测值，使用随机数流 "stream"；m 是一个整数，mean 为浮点型参数，stream 是一个整数。这个分布将在第 2.7 节中讨论。
- lcgrand(stream)　这是一个由 Simlib 使用的随机数发生器，该函数根据流 "stream"(整数型参数)返回 0 与 1 之间的(连续)均匀分布的一个浮点型观测值。其编码在附录 7A 的图 7.5 给出，而不在附录 2A 中。使用 Simlib 时，特别是引用(#including)头文件 simlib.h 时，不必像附录 7A 的图 7.5 那样写明 "#include lcgrand.h"，因为 simlib.h 本身包含了所需要的声明。
- lcgrandst(zset，stream)　该函数把流 "stream" 的随机数种子 "设置" 为长整型参数 zset。它在附录 7A 的图 7.5 给出。
- lcgrandgt(stream)　该函数返回流 stream 的随机数发生器的当前整数值长整型数，它在附录 7A 的图 7.5 给出，并在第 7 章中更详细地进行讨论。也可能用于从仿真停止的地方重启后续仿真(利用 lcgranst)，只要与使用随机数有关的话。

到此完成了 Simlib 的介绍，但在进一步介绍它应用的具体例子之前，我们用一个综述来小结本节，就是在仿真中如何进行典型的使用 Simlib 的组件。这就是，用户仍然要写一个 C 主函数及事件函数，但是 Simlib 的函数可以使编码更加容易。首先，我们必须确定事件，并决定什么样的表用于什么样的目的；表的编号及其属性的编号有很大的随意性，但必须保持一致。而且，任何用到的 sampst 和 timest 变量必须定义，随机数流也应一样。除了由 Simlib 定义的全局变量(通过头文件 simlib.h)外，用户通常需要声明整个模型的全局变量，也许包括局部变量。在主函数中，粗略地说，按列出顺序发生以下活动：

(1)读/写(确认)输入参数。

(2)调用 init_simlib 来初始化 Simlib 变量。

(3)(如果需要)对那些需要按照某一特定属性的值保持某类排序的表，设置 list_rank[list]为属性数，表 "list" 根据该属性数排序(如果除事件表外无其他表需要排序，则跳过这一步骤)。

(4)设置 maxatr 为任何表中所用到的属性的最大值。注意：为使 Simlib 正常运行，maxatr 必须至少设为 4。如果跳过这一步骤，maxatr 的默认值为 10，且仿真仍将正确运行，但是设置 maxatr 为一个较小的值将使仿真运行更快，因为它避免了重复拷贝未用属性进出表。

(5)(如果需要)调用 timest 来初始化所有初始值非零的 timest 变量。

(6)为安排在时刻 0 每个事件，调用 event_schedule 来初始化事件表。如果数据结构中所用的事件属性超过第一个(事件时间)和第二个(事件类型)属性，在为该事件调用 event_schedule 之前，用户有责任设置 transfer[3]、transfer[4]等。不可能发生的事件只要不放到事件表中就可。

(7)调用 timing 来确定 next_event_type，并将 sim_time 更新为该事件时间。

(8)按照 next_event_type 所确定的，调用相应的事件函数(用户编写的但在可能的地方用到 Simlib 的变量及函数)。这是一种典型的做法，用 case 语句，将控制转到几个事件函数调用语句的一个，如第 1 章在 C 程序中所做的那样。

(9)当仿真结束时，调用报告生成函数(用户编写)，该函数依次将调用 sampst、timest 或 filest，然后输出所要求的求和统计。另一种做法，报告生成器可调用 out_sampst、out_timest或 out_filest，按照标准格式输出求和统计。

在仿真运行时，通过使用 list_file 和 list_remove，加上 tranfer 数组传送进出表的记录的属性中的数据，实现表的维护。需要时，还可用 sampst 和 timest 来获得所关心的变量的统计信息。

关于 Simlib 能力的最后一点是有关错误检查。虽然没有软件包能够检测到所有错误并提供搞定错误的建议，但仍有一些专门的机会在仿真程序中做这件事，如在第 1 章中讨论的那样。因此，Simlib 包含了几个这样的检错功能，并输出一条信息(标准输出格式)指明错误的性质以及其发生时仿真钟的值。例如，Simlib 定时函数检测“时间颠倒”，即试图安排一个比当前更早时间发生的事件。此外，还有一些非法表数、非法变量数的检测，以及试图从空表中删除一个记录的检测，等等。

在第 2.4 节至第 2.7 节中我们将说明如何利用 Simlib 来仿真各种复杂度的系统。

2.4 单服务台排队系统的 Simlib 仿真

2.4.1 问题描述

本节我们将展示如何用 Simlib 仿真第 1.4 节中的单服务台排队系统。模型是完全一样的，因此我们可将注意力集中在 Simlib 的用法而不必担心新的模型结构。我们将使用 1 000个延误结束规则，如原来第 1.4.3 小节中的描述那样。

2.4.2 Simlib 程序

第一步是确定事件：与原来一样——到达事件为类型 1 事件，离去事件(完成服务)为类型 2 事件。

下一步，我们必须定义 Simlib 表及其记录中的属性。把这个写下来很重要，因为在开发程序时将要参考表 2.7 所示的属性。

表 2.7

表	属性 1	属性 2
1，队列	到达队列的时间	—
2，服务台	—	—
3，事件表	事件时间	事件类型

注意表 1(表示队列)非常类似于第 1.4.4 小节中用到的数组 time_arrival，区别仅在于现在我们利用了 Simlib 的表处理功能；表 1 只有一个属性。表 2 表示服务台，要么为空(服务台空闲)，要么只有一条记录(服务台忙碌)；当服务台忙碌时，表 2 中的记录是一个“虚拟”的记录，没有任何实际属性。定义这样一个表的目的是可允许在仿真结束时使用 filest 获得服务台利用率。另外，可以通过观察 list_size[2]是否为 1 来得知服务台是否忙碌。使用虚拟表可以便于取消变量 server_status，以及在仿真过程中或结束时我们无需使用 timest 来获得利用率。然而这并不是最高效的计算方法，因为一旦服务台改变状态，链表的所有机制都被调用，而并不仅仅是改变变量 server_status 的值(这是计算时间与分析者用于模型编码的时间这两者之间进行折中的好例子)。最后，表 25 为事件表，属性 1 表示事件时间，属性 2 表示事件类型；这在所有 Simlib 程序中都需要，但对于某些模型，

表25中的事件记录我们还会使用其他属性。

再下一步，我们应确定所有要用到的sampst和timest变量。这里我们只有如表2.8所示的sampst变量编号。

表 2.8

sampst 变量编号	含义
1	队列中的延误

既然可以通过filest(或者out_filest，若是标准输出格式的话)获得队列中的顾客数以及利用率统计，对本模型我们无需任何timest变量。

最后，我们分配独立的随机数流来生成到达间隔时间和服务时间，如表2.9所示。

表 2.9

流	用途
1	到达间隔时间
2	服务时间

图2.6给出了外部声明，出现在mm1smlb.c文件的开头。我们做的第一件事情是引用(#include)头文件Simlib.h，这对所有使用Simlib的程序都需要。为使编码更具可读性和通用性，我们将事件类型、表的编号、sampst变量，以及随机数流都定义成符号常量。我们还需声明一些模型的非simlib变量，但远比第1.4.4小节中的少，因为很多信息都在simlib内部。此外我们还需有我们自己的函数：实现本模型初始化函数init_model，到达事件函数arrive和离去事件函数depart，以及报告生成器report，但是我们不再需要定时函数或者expon函数，因为这些都由simlib提供。

```
/* External definitions for single-server queueing system using simlib. */

#include "simlib.h"              /* Required for use of simlib.c. */

#define EVENT_ARRIVAL         1  /* Event type for arrival. */
#define EVENT_DEPARTURE       2  /* Event type for departure. */
#define LIST_QUEUE            1  /* List number for queue. */
#define LIST_SERVER           2  /* List number for server. */
#define SAMPST_DELAYS         1  /* sampst variable for delays in queue. */
#define STREAM_INTERARRIVAL   1  /* Random-number stream for interarrivals. */
#define STREAM_SERVICE        2  /* Random-number stream for service times. */

/* Declare non-simlib global variables. */

int   num_custs_delayed, num_delays_required;
float mean_interarrival, mean_service;
FILE  *infile, *outfile;

/* Declare non-simlib functions. */

void init_model(void);
void arrive(void);
void depart(void);
void report(void);
```

图2.6 外部定义的C代码，Simlib的排队模型

图2.7给出了主函数，这个仍然必须由用户编写。打开输入、输出文件之后，我们读取输入参数，然后立即将其输出以确认其读取正确(并为形成我们的输出文件准备)。调用init_simlib对Simlib进行初始化，然后我们设置maxatr为4；尽管我们的所有记录都没有超过两个属性，但maxatr必须至少设置成4，以便Simlib正常运作。由于我们未使用任

何排序表(事件表除外)，因此我们无需在 list_rank 数组中设置任何东西；另外，两个连续时间函数(队列长度和服务台状态)均初始化为零，因此我们无需覆盖其默认值。然后调用用户自定义函数 init_model，以设置我们自己的、非 Simlib 的建模变量。主函数的其余部分类似于该模型的非 Simlib 版本的图 1.11，区别仅在于我们无需更新连续时间累加器，因为这由 Simlib 内部来处理。

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("mm1smlb.in",  "r");
    outfile = fopen("mm1smlb.out", "w");

    /* Read input parameters. */

    fscanf(infile, "%f %f %d", &mean_interarrival, &mean_service,
           &num_delays_required);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Single-server queueing system using simlib\n\n");
    fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
            mean_interarrival);
    fprintf(outfile, "Mean service time%16.3f minutes\n\n", mean_service);
    fprintf(outfile, "Number of customers%14d\n\n\n", num_delays_required);

    /* Initialize simlib */

    init_simlib();

    /* Set maxatr = max(maximum number of attributes per record, 4) */

    maxatr = 4;  /* NEVER SET maxatr TO BE SMALLER THAN 4. */

    /* Initialize the model. */

    init_model();

    /* Run the simulation while more delays are still needed. */

    while (num_custs_delayed < num_delays_required) {

        /* Determine the next event. */

        timing();

        /* Invoke the appropriate event function. */

        switch (next_event_type) {
            case EVENT_ARRIVAL:
                arrive();
                break;
            case EVENT_DEPARTURE:
                depart();
                break;
        }
    }

    /* Invoke the report generator and end the simulation. */

    report();

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

图 2.7 主函数的 C 代码，Simlib 的排队模型

图 2.8 给出了 init_model，其开始于将所观测的延误顾客数 num_custs_delayed 计数器设置为 0。通过调用 event_schedule 来安排第一个到达事件，将事件时间(浮点型)作为第一个参数，事件类型(整数型)作为第二个参数。注意：这里在第一个参数中将 sim_time 加到生成的指数到达间隔时间上并不是严格必需的，因为 sim_time 此时为零，但是我们这样写是为了展示其一般格式，以强调 event_schedule 的第一个参数是在仿真事件发生时的将来的(绝对)时间，而不是从现在到那时的间隔时间。在第 1 章中，我们设置不可能事件的时间为 $+\infty$(实际上是 10^{30})，但现在我们只要使其在事件表中不出现，就确保不会选择它们作为下一事件。因此，这里我们根本无需安排离去事件。

```
void init_model(void)  /* Initialization function. */
{
    num_custs_delayed = 0;

    event_schedule(sim_time + expon(mean_interarrival, STREAM_INTERARRIVAL),
                   EVENT_ARRIVAL);
}
```

图 2.8　函数 init_model 的 C 代码，Simlib 的排队模型

图 2.9 所示的为到达事件函数的代码，开始，使用 event_schedule 安排下一到达事件，方式类似于 init_model(这里，将 sim_time 加到生成的指数到达间隔时间上去是非常必要的，因为 sim_time 此时大于零)。然后我们需要检验服务台是否忙碌，即检查服务台表是否有(虚拟)记录；这由检查 list_size[LIST_SERVER]是否等于 1 来实现。如果为 1，到达的顾客必须加入队尾，即将其到达时间(当前仿真钟的值 sim_time)放到 transfer 数组的第一个位置，并将这条记录填入队列列表(list＝LIST_QUEUE＝1)的尾部(option＝LAST＝2)。注意，这里我们不必检验队列是否溢出，因为必要时 simlib 会为该表自动分配动态内存。另一方面，若服务台空闲，顾客的延误时间为 0，这是用调用 sampst 来表示；即使延误时间是 0，这个调用也是必要的，因为 sampst 将使观测数加 1。我们增加延误顾客数 num_custs_delayed 是因为一个延误被观测到了，且一个离去事件被安排到了事件表中；注意，我们现在用流 EVENT_DEPARTURE(＝2)来生成服务时间。

```
void arrive(void)  /* Arrival event function. */
{
    /* Schedule next arrival. */

    event_schedule(sim_time + expon(mean_interarrival, STREAM_INTERARRIVAL),
                   EVENT_ARRIVAL);

    /* Check to see whether server is busy (i.e., list SERVER contains a
       record). */

    if (list_size[LIST_SERVER] == 1) {

    /* Server is busy, so store time of arrival of arriving customer at end
       of list LIST_QUEUE. */

    transfer[1] = sim_time;
    list_file(LAST, LIST_QUEUE);
}

else {

    /* Server is idle, so start service on arriving customer, who has a
       delay of zero.  (The following statement IS necessary here.) */
```

图 2.9　函数 arrive 的 C 代码，Simlib 的排队模型

```
        sampst(0.0, SAMPST_DELAYS);

        /* Increment the number of customers delayed. */

        ++num_custs_delayed;

        /* Make server busy by filing a dummy record in list LIST_SERVER. */

        list_file(FIRST, LIST_SERVER);

        /* Schedule a departure (service completion). */

        event_schedule(sim_time + expon(mean_service, STREAM_SERVICE),
                       EVENT_DEPARTURE);
    }
}
```

图 2.9 (续)

图 2.10 中的事件函数 depart 通过观察队列表的长度来检查队列是否为空，队列表的长度保存在 Simlib 的 list_size[LIST_QUEUE]中。如果为空，这从表 LIST_SERVER 中删除(虚拟)记录，使服务台变为闲，只需这一动作。注意我们正在删除的是表的第一条记录，但也是删除了最后一条记录，因为只有一条记录。另一方面，如果有一个队列，其第一个顾客已从队列中删除，该顾客的到达时间通过 list_remove 已被存入 transfer[1]中。该顾客在队列中的延误时间为 sim_time-transfer[1]，在 sampst 中计算并保存，观测到的延误数增加；如第 1 章中的例子一样，如果仿真需要很长时间运行，则有必要定义 sim_time 和 transfer 为 double 类型，以避免在计算队列延误时间的减法中损失精度。最后，调用 event_schedule 以安排该顾客服务的完成。注意，这里我们不必将队列上移，这由 Simlib 使用例 2.1 中讨论的链表在内部处理。

```
void depart(void)  /* Departure event function. */
{
    /* Check to see whether queue is empty. */

    if (list_size[LIST_QUEUE] == 0)

        /* The queue is empty, so make the server idle and leave the departure
           (service completion) event out of the event list. (It is currently
           not in the event list, having just been removed by timing before
           coming here.) */

        list_remove(FIRST, LIST_SERVER);

    else {

        /* The queue is nonempty, so remove the first customer from the queue,
           register delay, increment the number of customers delayed, and
           schedule departure. */

        list_remove(FIRST, LIST_QUEUE);
        sampst(sim_time - transfer[1], SAMPST_DELAYS);
        ++num_custs_delayed;
        event_schedule(sim_time + expon(mean_service, STREAM_SERVICE),
                       EVENT_DEPARTURE);
    }
}
```

图 2.10 函数 depart 的 C 代码，Simlib 的排队模型

报告生成函数如图 2.11 所示，使用标准输出格式，队列延误时间用 out_sampst，而队列中顾客数及利用率用 out_files。注意，我们在调用 out_sampst 和 out_filest 之前加入一段简要开头，使报告稍微好读一些。

```
void report(void)  /* Report generator function. */
{
    /* Get and write out estimates of desired measures of performance. */

    fprintf(outfile, "\nDelays in queue, in minutes:\n");
    out_sampst(outfile, SAMPST_DELAYS, SAMPST_DELAYS);
    fprintf(outfile, "\nQueue length (1) and server utilization (2):\n");
    out_filest(outfile, LIST_QUEUE, LIST_SERVER);
    fprintf(outfile, "\nTime simulation ended:%12.3f minutes\n", sim_time);
}
```

图 2.11　函数 report 的 C 代码，Simlib 的排队模型

2.4.3 仿真输出与讨论

输出文件 mm1smlb.out 如图 2.12 所示，描述了由 out_sampst 和 out_filest 生成的标准输出格式。我们使用数字结果的通用格式来避免可能出现的域宽溢出。我们得到输出度量的所有特征值，即均值、最大值和最小值，还有 sampst 用到的离散时间变量的观测数。作为一个检验，我们还输出仿真钟的最终值。

```
Single-server queueing system using simlib

Mean interarrival time       1.000 minutes

Mean service time            0.500 minutes

Number of customers           1000


Delays in queue, in minutes:

  SAMPST                           Number
 variable                            of
  number        Average            values          Maximum          Minimum
 ___________________________________________________________________________

     1       0.5248728E+00     0.1000000E+04    0.5633087E+01    0.0000000E+00
 ___________________________________________________________________________


Queue length (1) and server utilization (2):

   File          Time
  number        average          Maximum          Minimum
 ___________________________________________________________

     1       0.5400774E+00    0.8000000E+01    0.0000000E+00

     2       0.5106925E+00    0.1000000E+01    0.0000000E+00
 ___________________________________________________________


Time simulation ended:      971.847 minutes
```

图 2.12　输出报告，Simlib 的排队模型

很重要的一点是，这些数值结果与同一模型的非 Simlib 版本在图 1.19 中的结果并不一样；事实上，是很不一样：队列的平均延误从第 1 章中的 0.430 变成了现在的 0.525，相差约 22%。其原因是我们现在使用“专用(dedicating)”的概念，使一个随机数流用于某一特定随机源，而在第 1 章中对所有的我们使用同一个流(编号 1)。两个程序都是正确的，而且这表明需要小心地统计分析仿真输出数据，如第 9 章至第 12 章中讨论的那样。

尽管使用 Simlib 并没有明显简化该模型的编码，但这个包拥有众多表结构，其价值在复杂模型中已经变得更为明显。类似模型将在下面的第 2.5 节至第 2.7 节中讨论。

2.5 分时计算机模型

本节我们用 Simlib 来仿真一个由 Adiri 和 Avi-Itzhak 提出(1969)的分时计算机设备的模型。

2.5.1 问题描述

一个公司有一个计算机系统，它由一个中央处理单元(CPU)和 n 个终端组成，如图 2.13所示。每个终端的操作员“思考”时间的数量是均值为 25 秒的指数随机变数，然后向 CPU 发送一个具有均值为 0.8 秒的指数分布的服务时间的作业。正在到达的作业加入 CPU 的单一队列，但按照轮转方式服务，而不是 FIFO 方式。这就是说，CPU 给每个作业分配一个最大服务额度，长度为 $q=0.1$ 秒。如果一个作业的(剩余)服务时间 s 秒不超过 q，则 CPU 花费 s 秒，外加一个固定的切换时间 $\tau=0.015$ 秒，来处理该作业，而后返回到它的终端。但是，如果 $s>q$，花费 $q+\tau$ 秒时间来处理这项作业，然后作业加入队尾，它的剩余服务时间要减去 q 秒。这个过程不断重复，直到最终完成该作业的服务，此时返回其终端，其操作员则开始下一次“思考”时间。

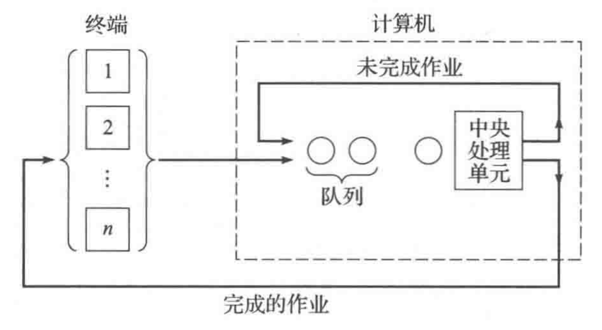


图 2.13 分时计算机模型

令 R_i 为第 i 个待完成服务的作业的响应时间，它定义为该作业离开其终端的瞬时到其完成 CPU 处理的瞬时之间历经的时间。对于 $n=10, 20, \cdots, 80$ 的每种情形，我们用 Simlib 来仿真计算机系统完成 1 000 个作业(1 000 个响应时间)，并估计这些作业的期望平均响应时间、队列中等待的作业的期望时间平均数，以及 CPU 的期望利用率。假设所有终端在时刻 0 时都处于“思考”状态。该公司想知道该系统可以有多少个终端，并仍能提供用户平均响应时间不超过 30 秒。

2.5.2 Simlib 程序

该模型的事件如表 2.10 所示。

表 2.10

事件描述	事件类型
arrival，“思考”时间结束时，作业从终端到达 CPU	1
end-CPU-run，一次 CPU 运行的结束，此时或者作业完成其服务需求，或者完成最大的处理额度 q	2
end-simulation，仿真结束	3

注意，我们已经定义了结束仿真事件，尽管这个模型的终止规则并不是一个固定的仿真时间点。结束仿真事件安排在观测到第 1 000 个响应时间的时刻，并安排在那一时刻立即发生。显然，这种终止准则还可以有其他方式实现，如下面讨论的那样。

该模型的一个事件图(见第 1.4.7 小节)如图 2.14 所示。arrival(即结束思考时间)事件的 n 个独立初始化指的是这样一个事实：n 个终端每个都需要初始安排这样一个事件。还要注意，arrival 事件与 end-CPU-run 事件之间存在互相调度的可能；如果正在到达的作业发现 CPU 空闲，则该到达可以调度 CPU 结束运行，而如果退出 CPU 的作业完成服务并返回其终端，则 end-CPU-run 事件可以调度一个到达。另外，有这样一种情况，即作业在整个处理完成之前离开 CPU 并循环返回队列，此时发现，由于所有其他作业在其终端处，因而队列空并且 CPU 闲置，只有此时，end-CPU-run 事件可以调度自己。最后注意，end-simulation 事件只能由 end-CPU-run 事件调度，时间开销为 0，在这种情况下，完成

的作业离开 CPU，并提供所需的最后一个(第 1 000 个)响应时间。正如第 1.4.7 小节中讨论的那样，带有细而平滑的入弧的事件(表示平滑箭头源处的事件以开销时间为 0 来调度该事件)可从模型中删除，其活动合并到别处。习题 2.2 涉及该模型的这种情形。

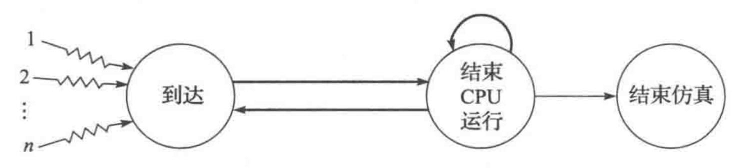


图 2.14 事件图、计算机模型

将用到三个记录表，一个对应队列中的作业(表 1)，一个对应 CPU 服务的作业(表 2)，一个为事件表(表 25，如通常做的那样)。这些表有如表 2.11 所示的属性。

表 2.11

表	属性 1	属性 2
1，队列	作业到达计算机的时间	剩余服务时间
2，CPU	作业到达计算机的时间	当前通过 CPU 之后的剩余服务时间(如果当前通过 CPU 是该作业所需的最后一次，则为负)
25，事件表	事件时间	事件类型

如第 2.4 节中的那样，我们用表来表示服务台(即表 2 表示 CPU)，因此可实现在仿真结束时通过 filest 估计 CPU 的利用率。然而，在这里，“服务台”表的属性并不是虚拟的；它们带有服务中作业的必要信息，因为一项作业可以重复访问 CPU 多次因而必须带上它的到达时间和剩余服务时间，以便正确地处理。另外还要注意，我们已经将表 1 和表 2 的属性进行了匹配，因此传送这两个表中的记录只要简单的调用 list_remove 和 list_file 就可以做到，而无需在 transfer 数组中重排这些属性。最后，我们并没有清晰地跟踪作业来自哪个终端，这样当其在计算机中完成处理时，我们不知道把它送回哪个终端以使该终端开始下一个思考时间。而现实中肯定不会这样，因为终端操作员会获取各自的输出返回，但在仿真中我们不需要通过特定终端来表示每项作业的属性，因为我们只想知道响应时间的总性能度量(包括所有终端)。进一步，所有终端在概率上都是同分布的，因为其思考时间和 CPU 时间的分布都是相同的。习题 2.3 丰富了该模型，要求收集各个终端各自的响应统计，并允许终端特性不同；这些变化的任何一个都需要当作业在计算机中时，它要携带作业源的终端信息。

由于只有一个感兴趣的离散时间统计(响应时间)，我们只需表 2.12 所示的一个 sampst 变量编号。

表 2.12

sampst 变量编号	含义
1	响应时间

对所要求的每个连续时间统计(队列中的作业个数及 CPU 利用率)，有一个相应的表，其长度表示所要求的量，因此我们可以再一次通过 filest 获得输出结果，并且我们不需要任何自己的 timest 变量。

该模型有两种类型的随机变数，我们使用表 2.13 所示的流分配。

表 2.13

流	用途
1	思考时间
2	服务时间

图 2.15 给出了本模型的全局外部定义，在 tscomp.c 文件开头。在引用(#including)头文件 simlib.h 之后，我们定义事件类型、表号、sampst 变量数，以及随机数流的符号常量，声明非 Simlib 变量，包括整型：仿真中终端数的最大值与最小值(min_terms=10，max_terms=80)、仿真中终端数的增量(incr_terms=10)、某一特定仿真中的终端数(num_terms)、当前仿真中观测到的响应时间的个数(num_responses)，以及需要的响应个数(num_responses_required=1 000)；浮点型的只需要对输入参数平均思考时间和平均服务时间，额度 q，及切换时间 τ 进行声明。声明用户自定义、非 Simlib 的函数，包括 arrival 与 end-CPU-run 事件的事件函数；我们已经编写了一个非事件函数 start_CPU_run 来处理 arrival 事件或 end-CPU-run 事件发生时可能发生的特定活动，以避免在这些事件函数中每个函数重复相同的代码块。没有独立的模型初始化函数，因为初始化模型的需求太少(simlib 在 init_simlib 中做的除外)，因此该活动只放入主函数中。对于本模型，我们选择不使用标准输出格式选项，因为我们实际要做 8 个独立仿真，并想将输出结果放到客户化的表中，每个仿真占一行；此外，我们只希望获得输出性能度量的均值，而不是它们的所有特性(如极值等)。

```
/* External definitions for time-shared computer model. */

#include "simlib.h"               /* Required for use of simlib.c. */

#define EVENT_ARRIVAL           1  /* Event type for arrival of job to CPU. */
#define EVENT_END_CPU_RUN       2  /* Event type for end of a CPU run. */
#define EVENT_END_SIMULATION    3  /* Event type for end of the simulation. */
#define LIST_QUEUE              1  /* List number for CPU queue. */
#define LIST_CPU                2  /* List number for CPU. */
#define SAMPST_RESPONSE_TIMES   1  /* sampst variable for response times. */
#define STREAM_THINK            1  /* Random-number stream for think times. */
#define STREAM_SERVICE          2  /* Random-number stream for service times. */

/* Declare non-simlib global variables. */

int   min_terms, max_terms, incr_terms, num_terms, num_responses,
      num_responses_required, term;
float mean_think, mean_service, quantum, swap;
FILE  *infile, *outfile;

/* Declare non-simlib functions. */

void arrive(void);
void start_CPU_run(void);
void end_CPU_run(void);
void report(void);
```

图 2.15　外部定义的 C 代码，计算机模型

主函数如图 2.16 所示。如通常那样，我们打开输入、输出文件，读取输入参数，并将其写入报告头。由于我们将做 8 个仿真，每个仿真有一输出行，我们此时也写出每列的头。一个 for 循环设置 num_terms 依次从 10，20，…，80，然后开始，包括主程序的其他部分，除了在快结束的地方关闭文件外。对于每个 num_terms 值，运行一次独立的仿真，包括初始化以及结果输出，都在这个 for 循环内。每个仿真通过调用 init_simlib 从一个全新的 simlib 初始化开始，然后设 maxatr 为 4，初始化 num_responses 为 0，以及对 num_terms 个终端的每个调用 event_schedule，来安排每个终端到 CPU 的第一个到达事件。注意，这样一来我们将 num_terms 个事件安排在事件表中，所有的类型均为 1，每个代表一个特定终端的初始思考时间。然后开始 do-while 循环，只要事件类型不是 end_simulation 事件，则调用定时函数与适当的事件函数；如果刚执行的事件是 end_simulation 事件(且已调用了 report)，那么 do-while 循环结束，从而我们根据终端数返回“for”循环入口。end_simulation 事件并不是最初安排好的，但是将会在完成第 1 000 个响应时间时在函数

end_CPU_run 中来安排，并在那时发生，于是主函数将调用函数 report 并结束当前仿真。当外层 for 循环结束时，运行了我们要求的所有仿真，包括产生每个仿真的输出，从而通过关闭所有文件，整个程序结束。

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("tscomp.in",  "r");
    outfile = fopen("tscomp.out", "w");

    /* Read input parameters. */

    fscanf(infile, "%d %d %d %d %f %f %f %f",
           &min_terms, &max_terms, &incr_terms, &num_responses_required,
           &mean_think, &mean_service, &quantum, &swap);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Time-shared computer model\n\n");
    fprintf(outfile, "Number of terminals%9d to%4d by %4d\n\n",
            min_terms, max_terms, incr_terms);
    fprintf(outfile, "Mean think time  %11.3f seconds\n\n", mean_think);
    fprintf(outfile, "Mean service time%11.3f seconds\n\n", mean_service);
    fprintf(outfile, "Quantum          %11.3f seconds\n\n", quantum);
    fprintf(outfile, "Swap time        %11.3f seconds\n\n", swap);
    fprintf(outfile, "Number of jobs processed%12d\n\n\n",
            num_responses_required);
    fprintf(outfile, "Number of      Average         Average");
    fprintf(outfile, "         Utilization\n");
    fprintf(outfile, "terminals   response time  number in queue     of CPU");

    /* Run the simulation varying the number of terminals. */

    for (num_terms = min_terms; num_terms <= max_terms;
         num_terms += incr_terms) {

        /* Initialize simlib */

        init_simlib();

        /* Set maxatr = max(maximum number of attributes per record, 4) */

        maxatr = 4;  /* NEVER SET maxatr TO BE SMALLER THAN 4. */

        /* Initialize the non-simlib statistical counter. */

        num_responses = 0;

        /* Schedule the first arrival to the CPU from each terminal. */

        for (term = 1; term <= num_terms; ++term)
            event_schedule(expon(mean_think, STREAM_THINK), EVENT_ARRIVAL);

        /* Run the simulation until it terminates after an end-simulation event
           (type EVENT_END_SIMULATION) occurs. */

        do {

            /* Determine the next event. */

            timing();

            /* Invoke the appropriate event function. */

            switch (next_event_type) {
```

图 2.16

```
                case EVENT_ARRIVAL:
                    arrive();
                    break;
                case EVENT_END_CPU_RUN:
                    end_CPU_run();
                    break;
                case EVENT_END_SIMULATION:
                    report();
                    break;
            }

        /* If the event just executed was not the end-simulation event (type
           EVENT_END_SIMULATION), continue simulating.  Otherwise, end the
           simulation. */

        } while (next_event_type != EVENT_END_SIMULATION);
    }

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

图 2.16 （续）

到达事件的流程图如图 2.17 所示，其代码如图 2.18 所示。当在计算机中(即在队列中或在 CPU 中)的时候，每项作业都有自己的记录，其属性描述如前。由于本事件表示在一个思考时间结束时一个作业到达计算机，其属性必须现在定义，因此到达时间存于第一属性，且生成总的服务需求并存于第二属性。然后这项新到达的作业记录置于队尾。然而，CPU 可能实际上是空闲的(即表 LIST_CPU 中的记录数 list_size[LIST_CPU]等于 0)，在这种情形下，调用函数 start_CPU_run 来从队中取出该项作业(这里可能是唯一的作业)并将其放入 CPU 中开始它的处理。该函数隐含的一个逻辑是，到达计算机发现 CPU 空闲的作业，不能直接进入，而是必须先进入队列然后立即出来；这其实是做此事的一个物理假设，因为每当作业离开队列进入 CPU 时，就招致一个切换时间，就像 start_CPU_run 函数执行的那样，这在下面讨论。

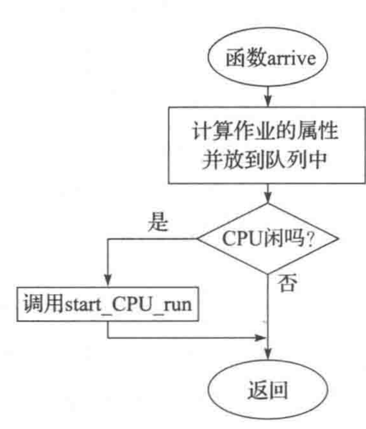


图 2.17 到达函数的流程图，计算机模型

```
void arrive(void)  /* Event function for arrival of job at CPU after think
                      time. */
{

    /* Place the arriving job at the end of the CPU queue.
       Note that the following attributes are stored for each job record:
            1. Time of arrival to the computer.
            2. The (remaining) CPU service time required (here equal to the
               total service time since the job is just arriving). */

    transfer[1] = sim_time;
    transfer[2] = expon(mean_service, STREAM_SERVICE);
    list_file(LAST, LIST_QUEUE);

    /* If the CPU is idle, start a CPU run. */

    if (list_size[LIST_CPU] == 0)
        start_CPU_run();
}
```

图 2.18 函数 arrive 的 C 代码，计算机模型

非事件函数 start_CPU_run 的流程图如图 2.19 所示，其代码如图 2.20 所示。该函数的设计是，正如刚刚讨论的，由事件函数 arrive 调用，或由事件函数 end_CPU_run 调用；因此它必须足够通用，以处理每种情形。该函数旨在将队首作业调出队，将其放入 CPU，并根据其要么完成服务要么用完整个额度来安排它离开 CPU 的时间。首先要做的是将作业从队首移出，通过调用 list_remove 来完成。然后，计算作业占用 CPU 的时间，该时间是额度时间和剩余服务时间的较小者(作业的第二个属性的值，刚刚由 list_remove 传送到 transfer[2]中)，再加上一个切换时间。在将作业的记录存入 CPU 表之前，其剩余服务时间(在 transfer[2]中)减去一个完整的额度时间，即使它只需要部分额度时间，在这种情况下该作业的第二个属性变成负数，作业通过后离开 CPU 刚刚开始时，我们用该条件作为该作业已经完成并返回其终端的标志。另一方面，如果作业通过 CPU 后还未完成，此时它将占用整个额度，并且其第二个属性将减去整个额度，从而正确地表示出在本次 CPU 通过后所需的(非负的)剩余服务时间。最后，调用 list_file 将作业放入 CPU 表中(注意：transfer[1]表示本作业到达计算机的时间，它已经被正确设置，因为它来自刚刚从队列表中删除的记录，其第一属性的定义与它是一样的)，接着结束这次 CPU 运行的事件，并被调度到事件表中。

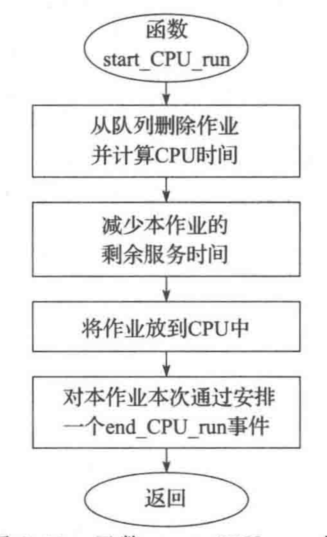


图 2.19 函数 start_CPU_run 的流程图，计算机模型

```
void start_CPU_run(void)  /* Non-event function to start a CPU run of a job. */
{
    float run_time;

    /* Remove the first job from the queue. */

    list_remove(FIRST, LIST_QUEUE);

    /* Determine the CPU time for this pass, including the swap time. */

    if (quantum < transfer[2])
        run_time = quantum + swap;
    else
        run_time = transfer[2] + swap;

    /* Decrement remaining CPU time by a full quantum.  (If less than a full
       quantum is needed, this attribute becomes negative.  This indicates that
       the job, after exiting the CPU for the current pass, will be done and is
       to be sent back to its terminal.) */

    transfer[2] -= quantum;

    /* Place the job into the CPU. */

    list_file(FIRST, LIST_CPU);

    /* Schedule the end of the CPU run. */

    event_schedule(sim_time + run_time, EVENT_END_CPU_RUN);
}
```

图 2.20 函数 start_CPU_run 的 C 代码，计算机模型

当一项作业完成一次通过 CPU 时，主函数调用事件函数 end_CPU_run；其流程图如图 2.21所示，代码在图 2.22 中列出。首先，将作业从 CPU 中删除，然后检查其是否仍需更多的 CPU 时间，即其第二个属性是否为正数。如果是，只需将其放回到队尾(注意队列

表与 CPU 表的属性匹配，使得 transfer 的内容是正确的)，调用 start_CPU_run 删除队首作业并开始它的处理。反之，如果 CPU 中出来的作业已完成，则它的响应时间 sim_time-transfer[1]由 sampst 寄存；与以前一样，对长时间仿真来说，sim_time 和 transfer 数组两者都必须设为 double 类型，以避免此时减法的精度损失。安排它的下一思考时间，增加观察到的响应时间数。然后检查该响应时间是否是最后一个所需要的；如果是，则安排 end-simulation 事件立即发生(传给 event_schedule 的第一个参数为 sim_time，当前仿真时间)，且定时函数将立即执行 end-simulation 事件(即不再传送任何仿真时间)，主函数调用报告函数来结束仿真。但是，如果仿真尚未结束，倘若队列非空，则调用 start_CPU_run；若队列是空的，则无动作发生，只是仿真继续。

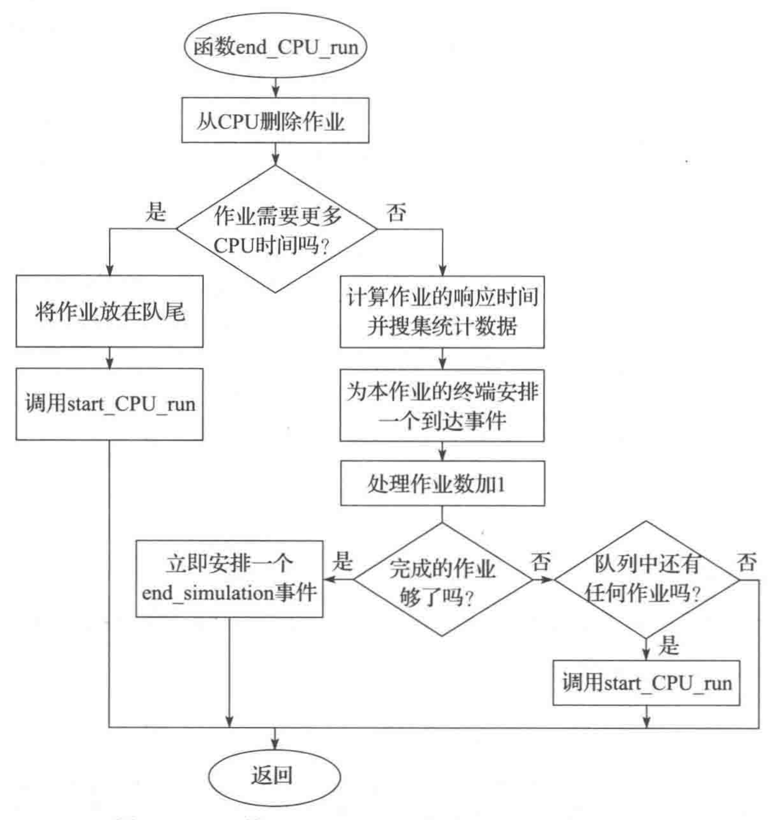


图 2.21 函数 end_CPU_run 的流程图，计算机模型

报告生成器在图 2.23 中列出，它只向输出文件中写入一行信息，包括刚完成仿真中的终端个数、由 sampst 返回的平均响应时间，以及由 filest 返回的事件平均队列长度和服务台利用率(回顾一下，sampst、timest 及 filest 除了将平均值存入 transfer[1]中外，还以它们的名字返回它们计算的平均值)。

2.5.3 仿真输出与讨论

输出文件 tscomp.out 如图 2.24 所示。如预想一样，计算机的拥挤度随着终端数的增加而变坏，这由平均响应时间、平均队列长度及 CPU 使用率来度量。特别是，我们看到，该系统可承受 60 个左右终端，其平均响应时间将不会降到远坏于 30 秒。在这一水平，我们可以看出平均队列长度大约为 30 个作业，这对确定容纳这些作业所需空间的总量是有用的(对于此目的，最大队列长度也许是更好的信息)；另外，在这样一个系统中 CPU 几乎时刻处于忙碌状态。然而，对于这些结论，我们通常要告诫大家：结论所基于的输出数据只是从系统一次运行的结果(可能任意长度)得到的，因此其精确度是未知的。

```
void end_CPU_run(void)  /* Event function to end a CPU run of a job. */
{
    /* Remove the job from the CPU. */

    list_remove(FIRST, LIST_CPU);

    /* Check to see whether this job requires more CPU time. */

    if (transfer[2] > 0.0) {

        /* This job requires more CPU time, so place it at the end of the queue
           and start the first job in the queue. */

        list_file(LAST, LIST_QUEUE);
        start_CPU_run();
    }

    else {

        /* This job is finished, so collect response-time statistics and send it
           back to its terminal, i.e., schedule another arrival from the same
           terminal. */

        sampst(sim_time - transfer[1], SAMPST_RESPONSE_TIMES);

        event_schedule(sim_time + expon(mean_think, STREAM_THINK),
                       EVENT_ARRIVAL);

        /* Increment the number of completed jobs. */

        ++num_responses;

        /* Check to see whether enough jobs are done. */

        if (num_responses >= num_responses_required)

            /* Enough jobs are done, so schedule the end of the simulation
               immediately (forcing it to the head of the event list). */

            event_schedule(sim_time, EVENT_END_SIMULATION);

        else

            /* Not enough jobs are done; if the queue is not empty, start
               another job. */

            if (list_size[LIST_QUEUE] > 0)
                start_CPU_run();
    }
}
```

图 2.22 函数 end_CPU_run 的 C 代码，计算机模型

```
void report(void)  /* Report generator function. */
{
    /* Get and write out estimates of desired measures of performance. */

    fprintf(outfile, "\n\n%5d%16.3f%16.3f%16.3f", num_terms,
            sampst(0.0, -SAMPST_RESPONSE_TIMES), filest(LIST_QUEUE),
            filest(LIST_CPU));
}
```

图 2.23 函数 report 的 C 代码，计算机模型

```
Time-shared computer model

Number of terminals        10 to  80 by   10

Mean think time        25.000 seconds

Mean service time       0.800 seconds

Quantum                 0.100 seconds

Swap time               0.015 seconds

Number of jobs processed         1000
```

Number of terminals	Average response time	Average number in queue	Utilization of CPU
10	1.324	0.156	0.358
20	2.165	0.929	0.658
30	5.505	4.453	0.914
40	12.698	12.904	0.998
50	24.593	23.871	0.998
60	31.712	32.958	1.000
70	42.310	42.666	0.999
80	47.547	51.158	1.000

图 2.24 输出报告，计算机模型

2.6 可换队的多出纳台银行

我们现在用 Simlib 来仿真一个多出纳台银行系统，其中允许顾客从一个队列中换(移动)到另一个队列中，只要对他们有利的话。这个模型还说明如何处理另一种常见的仿真终止规则。

2.6.1 问题描述

银行有 5 个出纳台，上午 9 点开门，下午 5 点关门，但营业直到到 5 点之前的所有顾客都完成服务为止。假设顾客的到达间隔时间是 IID(独立同分布)指数随机变量，均值为 1 分钟；并假设顾客的服务时间也是 IID 指数随机变量，均值为 4.5 分钟。

每个出纳台有独自的队列。到达顾客都加入到最短的队列中，如果同时有两个或以上的最短队列，则加入到最左边的队列中。令 n_i 为在某一特定时刻出纳台 i 前的顾客总数(包括队列中的顾客，以及服务中的顾客，如果有的话)。如果出纳台 i 的顾客服务的完成导致 $n_j > n_i + 1$，j 为另一出纳台，那么队列 j 尾部的顾客将换到队列 i 的尾部(如果有两个或以上这样的顾客，则换到最靠近的、最左边的那个队列)。如果出纳台 i 空闲，则换队顾客在出纳台 i 开始服务；参见图 2.25。

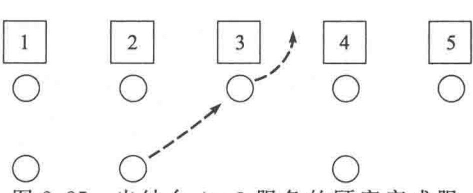


图 2.25 出纳台 $i=3$ 服务的顾客完成服务，引起一个顾客队列 $j=2$ 尾的顾客换队

银行经理关心运营成本以及当前向顾客提供的服务的质量，并考虑改变出纳台的个数。对于 $n=4$，5，6 和 7 个出纳台的每种情形，我们用 Simlib 仿真该银行系统，并估计队列中顾客的期望时间平均总数、队列中期望平均延误时间，以

及队列中期望最大延误时间。在所有情形中均假设银行开门时没有顾客。

2.6.2 Simlib 程序

本模型的事件如表 2.14 所示。

表 2.14

事件描述	事件类型
顾客到达银行	1
顾客完成服务后离去	2
银行下午 5 点关门	3

该模型的事件图在图 2.26 中给出。该图与固定运行长度的单服务台队列的相同(参见图 1.27)，只是结束仿真事件由关门事件代替。尽管这两个事件以相同的方式放到在事件图中，但它们产生的行为相当不同。

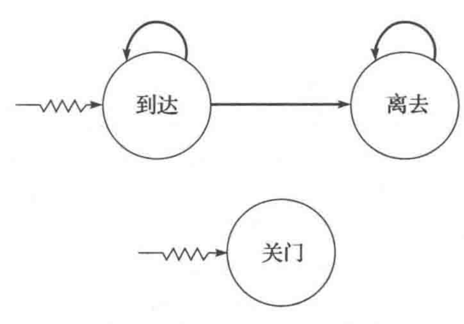


图 2.26 事件图，银行模型

本模型需要 $2n+1$ 个记录表，其中 n 为某次仿真运行的出纳台个数。表 $1\sim n$ 包含每个队列中等待的顾客的记录。表 $n+1\sim 2n$ 用于标明出纳台是否忙。如果表 $n+i$(其中 $i=1, 2, \cdots, n$)包含一条记录，则出纳台 i 忙；如果不包含记录，则出纳台 i 闲。最后，如通常那样，表 25 为事件列表。所有这些表的属性如表 2.15 所示。

表 2.15

表	属性 1	属性 2	属性 3
$1\sim n$，队列	到达队列的时间	—	—
$n+1\sim 2n$，出纳台	—	—	—
25，事件表	事件时间	事件类型	出纳台编号，若事件类型为 2

这里，在本模型中我们对服务台又用到了单独表；在这种情形下，这样做的唯一原因是为表达服务台的忙/闲状态，因为这些表记录的属性中没有有意义的信息，而且我们未要求进行服务台利用率的统计。还要注意，我们利用了在事件表记录中存储的信息不只是事件时间和事件类型这个机会。这样做的原因是在离去事件(类型 2)的情形下，我们需要知道离去发生的出纳台编号，以便正确地管理队列和换队规则。这也暗示我们在编程时必须记得：在为类型 2 事件调用 Simlib 函数 event_schedule 之前，先要给 transfer[3]定义一个值，因为在将事件记录存入事件表前，event_schedule 只把属性拷贝到 transfer[1]和 transfer[2]中。

本模型中的统计收集也有点不同。因为有多个不同的队列，因而，有换队的可能性，一个顾客可能在几个不同队列中经历他或她的延误(在到达系统与在某个服务台开始服务之间所花时间)。不管顾客可能在哪个队列，他或她都携带其到达时间(队列表中的属性 1)，使得在服务开始时可以计算延误时间。因此，我们只需简单地把所有顾客的延误时间都加在一起放到 sampst 变量中，如表 2.16 所示。

表 2.16

sampst 变量编号	含义
1	队列中的延误时间

sampst 还可以使我们能自动获得队列(或多个队列)中的最大延误时间。

我们还希望得到队列中顾客的时间平均总数，计算如下。我们令 $Q_i(t)$ 为在时间 t 时

队列 $i(i=1, 2, \cdots, n)$中的顾客数，则

$$Q(t) = \sum_{i=1}^{n} Q_i(t) \tag{2.1}$$

为在时间 t 时所有队列中的顾客总数。因此，我们想计算的是：

$$\hat{q} = \frac{\int_0^T Q(t)\,dt}{T} \tag{2.2}$$

其中，T 是仿真结束的时间(由如上描述的终止规则决定)。

然而，我们将式(2.1)代入式(2.2)，并利用积分的线性，我们得到

$$\hat{q} = \hat{q}_1 + \hat{q}_2 + \cdots + \hat{q}_n$$

其中，

$$\hat{q}_i = \frac{\int_0^T Q_i(t)\,dt}{T}$$

是队列 i 中顾客个数的时间平均值。

所有这些实际上说明，各个队列长度总和的平均等于它们平均长度的总和。因此，我们可用 filest(用于各个队列的表)在仿真结束时获得各个$\hat{q}_i$，并只要将它们加起来即可得到$\hat{q}$。自然，$\hat{q}$可通过定义与$Q(t)$对应的 timest 变量直接获得，每次到达时累加，每次服务开始时减少；但是，无论如何，我们必须保留这些队列表，所以仍倾向于使用前述方法(习题 2.4 考虑了该问题的一个引申，该题中我们需要知道队列中顾客总数的最大值以及以上各项统计；提出的问题是：总数的最大值是否为各个最大值的总和)。

本模型中有两类随机变量：到达间隔时间和服务时间。我们使用表 2.17 所示的流分配。

表 2.17

流	用途
1	到达间隔时间
2	服务时间

图 2.27 给出了本模型的外部定义及全局变量。按照通常的做法，我们引用(#include)Simlib 的头文件 simlib.h，再定义符号常量以用于事件类型、sampst 变量，以

```
/* External definitions for multiteller bank. */

#include "simlib.h"               /* Required for use of simlib.c. */

#define EVENT_ARRIVAL          1  /* Event type for arrival of a customer. */
#define EVENT_DEPARTURE        2  /* Event type for departure of a customer. */
#define EVENT_CLOSE_DOORS      3  /* Event type for closing doors at 5 P.M. */
#define SAMPST_DELAYS          1  /* sampst variable for delays in queue(s). */
#define STREAM_INTERARRIVAL    1  /* Random-number stream for interarrivals. */
#define STREAM_SERVICE         2  /* Random-number stream for service times. */

/* Declare non-simlib global variables. */

int   min_tellers, max_tellers, num_tellers, shortest_length, shortest_queue;
float mean_interarrival, mean_service, length_doors_open;
FILE  *infile, *outfile;

/* Declare non-simlib functions. */

void arrive(void);
void depart(int teller);
void jockey(int teller);
void report(void);
```

图 2.27　外部定义的 C 代码，银行模型

及随机数流号。下一步，声明整型参数以用于出纳台个数的最小值和最大值(分别为4和7)，表示我们将要进行仿真的各个情形，以及某一特定仿真的出纳台个数；“短”整型参数用于到达顾客的选队决策。输入参数声明为浮点型；length_doors_open假设以小时为单位读入，而仿真中其他地方用到的时间单位为分钟，为此编码时必须进行调整。函数都原型化了，到达为类型1事件，离去为类型2事件(带有一个整型参数表示离去发生的出纳台编号)，换队为一个非事件函数，带有一个整数型参数——服务完成所在出纳台编号(因此是换队顾客可能的目的地)，而报告函数输出仿真在下午5点或之后结束时的结果。

主函数如图2.28所示，一开始是打开输入及输出文件，读取输入参数，将其写回到输出文件，并生成报告头。如同计算机模型中一样，主函数的大部分外层为一个for循环，索引num_tellers表示当前模型变量出纳台个数n。调用init_simlib进行Simlib初始化(注意每个模型变量都必须做这件事，因此位于for循环内部)，设置maxatr为4(在我们的任意记录中，属性个数均不超过3，但对Simlib来说只能设maxatr不小于4才能正常运行)。调度第一个到达，还要安排关门事件，注意将时间单位变换为分钟。然后while循环开始，继续运行当前仿真，直到事件表为空，然后，当前的仿真结束。需要说明一下为何这是实现本模型终止规则的一种有效方式。只要是在下午5点之前，关门事件就一直在事件表中，且总会有安排下一个到达，因此事件表不会空。在5点时关门事件将发生，将其从事件表中删除该事件记录，该事件的作用是删除(用Simlib的函数event_cancel)下一到达事件，从而，从那时起，“阻止”了到达流，并从事件表中删除该事件(因为到达时间只有在初始化及处理到达时才被安排)。此时唯一的其他类型的事件为离去事件；且如果5点时银行里还有顾客，他们当中一些正在服务中，他们的离去事件均在事件表中。最终(或者在5点时也许马上，如果银行里此时正好没有顾客的话)，银行中的所有顾客都将接受服务并离去，此时事件表变成空，仿真将结束。只要while循环还在执行，我们按常规调用定时函数，将控制传给到达事件或离去事件的arrive或depart函数(注意，传给depart函数的整数型参数cast是出纳台个数，为离去事件记录的第三个属性)；如上面所描述的那样，当变成关门时间时，则执行close-door事件。当while循环结束时，当前仿真结束，调用report函数以产生输出。for循环结束后，所有仿真结束，从而我们关闭输入和输出文件，终止程序。

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("mtbank.in",  "r");
    outfile = fopen("mtbank.out", "w");

    /* Read input parameters. */

    fscanf(infile, "%d %d %f %f %f", &min_tellers, &max_tellers,
           &mean_interarrival, &mean_service, &length_doors_open);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Multiteller bank with separate queues & jockeying\n\n");
    fprintf(outfile, "Number of tellers%16d to%3d\n\n",
            min_tellers, max_tellers);
    fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
            mean_interarrival);
    fprintf(outfile, "Mean service time%16.3f minutes\n\n", mean_service);
    fprintf(outfile,
            "Bank closes after%16.3f hours\n\n\n\n", length_doors_open);

    /* Run the simulation varying the number of tellers. */
```

图2.28 主函数的C代码，银行模型

```
    for (num_tellers = min_tellers; num_tellers <= max_tellers; ++num_tellers) {

        /* Initialize simlib */

        init_simlib();

        /* Set maxatr = max(maximum number of attributes per record, 4) */

        maxatr = 4;  /* NEVER SET maxatr TO BE SMALLER THAN 4. */

        /* Schedule the first arrival. */

        event_schedule(expon(mean_interarrival, STREAM_INTERARRIVAL),
                       EVENT_ARRIVAL);

        /* Schedule the bank closing.  (Note need for consistency of units.) */

        event_schedule(60 * length_doors_open, EVENT_CLOSE_DOORS);

        /* Run the simulation while the event list is not empty. */

        while (list_size[LIST_EVENT] != 0) {

            /* Determine the next event. */

            timing();

            /* Invoke the appropriate event function. */

            switch (next_event_type) {

                case EVENT_ARRIVAL:
                    arrive();
                    break;
                case EVENT_DEPARTURE:
                    depart((int) transfer[3]);  /* transfer[3] is teller
                                                   number. */
                    break;
                case EVENT_CLOSE_DOORS:
                    event_cancel(EVENT_ARRIVAL);
                    break;
            }
        }

        /* Report results for the simulation with num_tellers tellers. */

        report();
    }

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

图 2.28 （续）

到达事件的流程图和代码在图 2.29 和图 2.30 中给出。函数开始于调度下一到达事件。然后开始一个 for 循环，随着索引变量 teller 按出纳台编号改变，依次观察每个出纳台(表编号为 $n+1$，$n+2$，…，$2n$)，看它们是否空闲(即 list_size[num_tellers+teller]是否为 0)。只要发现一个出纳台空闲，就在 sampst 中登记一个顾客的 0 延误时间，将一条虚拟记录放到该出纳台表中，该出纳台就置成忙，并安排该顾客的服务完成事件。然后，return 语句将控制传回给主函数，既无到达函数的剩余部分的执行，也无该循环的剩余部分(如果有的话)的执行。到达函数的其余部分是指所有出纳台都忙碌的情形，循环的剩余部分是指其他编号更大的出纳台，这些出纳台在任何情况下我们不用观察，因为最小编号的空闲出纳台优先。如果这个 for 循环完成，则所有出纳台忙碌，下一个 for 循环进行跨队列搜索，以发现最短队列；如果有结的话，则确定最小编号的队列。该解结规则由该 for 循环内 if 语句的严格不等式(<)来执行，其含义是，在从左到右的搜索中，只有当一

个新的队列严格短于之前的选择时，它才能选择该新队列。完成该 for 循环之后，整型参数 shortest_queue 将表示所选队列的编号，该到达的顾客进入该队列队尾，其到达时间为唯一所需属性。

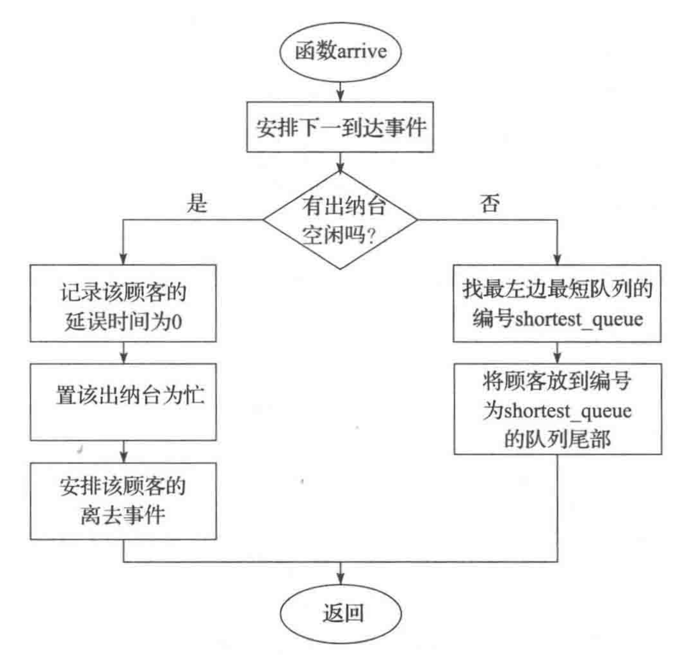


图 2.29　到达函数的流程图，银行模型

```
void arrive(void)  /* Event function for arrival of a customer to the bank. */
{
    int teller;

    /* Schedule next arrival. */

    event_schedule(sim_time + expon(mean_interarrival, STREAM_INTERARRIVAL),
                   EVENT_ARRIVAL);

    /* If a teller is idle, start service on the arriving customer. */

    for (teller = 1; teller <= num_tellers; ++teller) {

        if (list_size[num_tellers + teller] == 0) {

            /* This teller is idle, so customer has delay of zero. */

            sampst(0.0, SAMPST_DELAYS);

            /* Make this teller busy (attributes are irrelevant). */

            list_file(FIRST, num_tellers + teller);

            /* Schedule a service completion. */

            transfer[3] = teller;  /* Define third attribute of type-two event-
                                      list record before event_schedule. */

            event_schedule(sim_time + expon(mean_service, STREAM_SERVICE),
                           EVENT_DEPARTURE);

            /* Return control to the main function. */
```

图 2.30　函数 arrive 的 C 代码，银行模型

```
            return;
        }
    }

    /* All tellers are busy, so find the shortest queue (leftmost shortest in
       case of ties). */

    shortest_length = list_size[1];
    shortest_queue  = 1;
    for (teller = 2; teller <= num_tellers; ++teller)
        if (list_size[teller] < shortest_length) {
            shortest_length = list_size[teller];
            shortest_queue  = teller;
        }

    /* Place the customer at the end of the leftmost shortest queue. */

    transfer[1] = sim_time;
    list_file(LAST, shortest_queue);
}
```

图 2.30 （续）

事件函数 depart 的流程图和代码在图 2.31 和图 2.32 中给出。当一个顾客完成服务时，该函数由主程序调用；整型参数 teller 为完成服务的出纳台的编号。如果该出纳台的队列为空(list_size[teller]为 0)，通过删除相应列表中的虚拟记录，则该出纳台置成闲，调用函数 jockey 来决定来自其他队列中的顾客是否能换队在编号为“teller”的出纳台上服务，该出纳台刚刚变成闲。另一方面，如果该出纳台的队列非空，则将第一个顾客移出，在 sampst 中记录他或她的队列中的延误时间(sim_time-transfer[1])，并安排服务完成事件；对于长仿真，sim_time 和 transfer 必须定义为 double 类型，以避免计算队列中延误时间做减法时损失精度。注意，我们有责任在调用 event_schedule 之前定义 transfer[3]为出纳台编号，因为在将其存入事件表之前，本函数只将事件的时间和类型(属性 1 和 2)拷贝到 transfer 数组中。在这种情况下，我们必须调用 jockey 函数来检查是否有来自其他队列的顾客想换到该队列中(到达发生后应该无顾客换队，因为这并不能减少顾客期望的离去时间)。

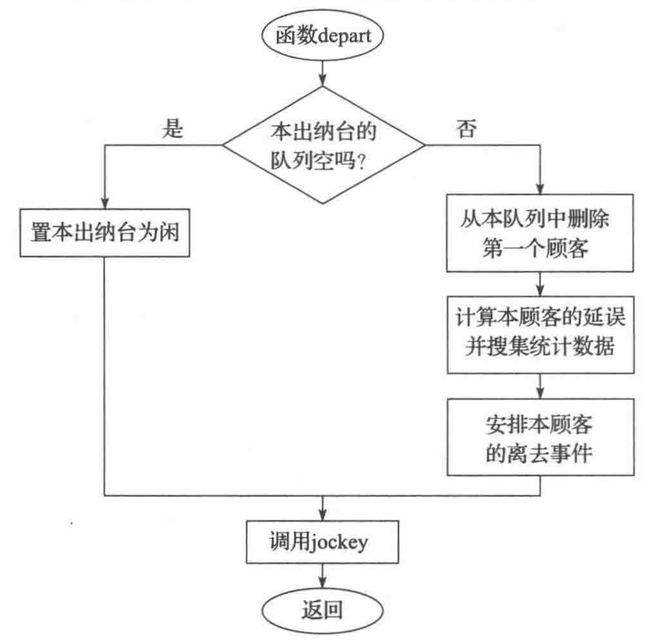


图 2.31 离去函数的流程图，银行模型

```
void depart(int teller)  /* Departure event function. */
{
    /* Check to see whether the queue for teller "teller" is empty. */

    if (list_size[teller] == 0)

        /* The queue is empty, so make the teller idle. */

        list_remove(FIRST, num_tellers + teller);

    else {

        /* The queue is not empty, so start service on a customer. */

        list_remove(FIRST, teller);
        sampst(sim_time - transfer[1], SAMPST_DELAYS);
        transfer[3] = teller;  /* Define before event_schedule. */
        event_schedule(sim_time + expon(mean_service, STREAM_SERVICE),
                       EVENT_DEPARTURE);
    }

    /* Let a customer from the end of another queue jockey to the end of this
       queue, if possible. */

    jockey(teller);
}
```

图 2.32　函数 depart 的 C 代码，银行模型

非事件函数 jockey 用整型参数 teller 调用，以观察一个顾客是否能从另一个(更长的)队列换到出纳台 teller 的队列中，或可能直接进入出纳台 teller 服务，若它刚刚变为空闲的话。其流程图与代码如图 2.33 和图 2.34 所示。整型变量 jumper 用于保留换队顾客(如

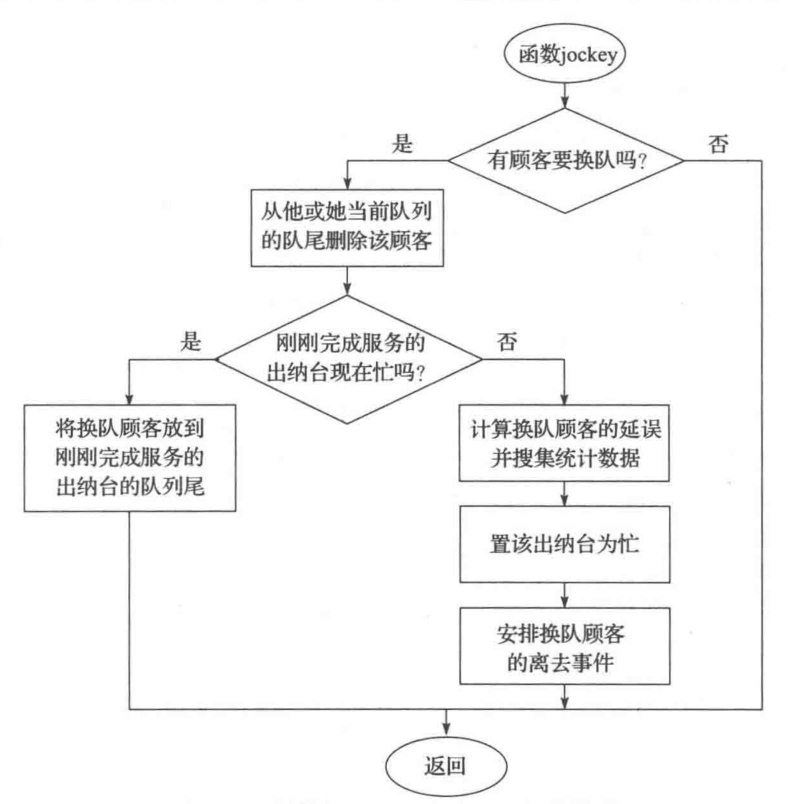


图 2.33　函数 jockey 的流程图，银行模型

果有的话)的队列编号；它初始化为零，且只有发现有换队顾客时才置为正数。整型变量 min_distance 是潜在的换队者到目的队列的(绝对)距离(用队列数表示)，它初始化为一个很大的值，因为我们希望搜索最短的这种距离。面对出纳台 teller 的顾客数是一个整数型变量 ni，即 ni$=n_i$，$i=$“teller”。for 循环检查队列(other_teller)看其是否满足换队要求，这里用条件 other_teller≠teller(因为顾客不会换到自己所处队列)且 nj＞ni＋1，其中，nj 为面对 other_teller 出纳台的顾客数，即 nj$=n_j$，$j=$other_teller。如果两个条件均满足，那么在队列编号为 other_teller 的队尾的顾客会希望换队，且如果她比稍前的也希望换队的顾客严格地更靠近目标队列的话(也就是，如果当前希望换队者相距目标队列的队列数即变量 distance 严格小于稍前最靠近的希望换队者的距离)，该顾客将(可能是临时地)被告知换队通过。注意，在有两个最靠近的希望换队者的情况下(一个在左，一个在右)，我们将换左边这个，因为右边的一个必须是严格地更靠近。当该 for 循环结束时，如果其他队列的长度满足无顾客想要换队，那么 jumper 变成零，此时控制回到主函数，无任何动作。然而，如果 jumper 是正数，则它等于将要换队的顾客所在队列编号，该顾客从其队尾删除。然后进行检查，看看刚完成服务的出纳台是否忙碌(忙则有顾客，该顾客是该出纳台队列的第一个)，在这种情况下，换队顾客只好加入他的新队列的队尾。但是，如果该出纳台空闲，那么换队顾客直接进入服务，计算并保存它的延误时间，服务台重新置为忙，安排换队顾客的服务完成。

```
void jockey(int teller)  /* Jockey a customer to the end of queue "teller" from
                            the end of another queue, if possible. */
{
    int jumper, min_distance, ni, nj, other_teller, distance;

    /* Find the number, jumper, of the queue whose last customer will jockey to
       queue or teller "teller", if there is such a customer. */

    jumper       = 0;
    min_distance = 1000;
    ni           = list_size[teller] + list_size[num_tellers + teller];

    /* Scan all the queues from left to right. */

    for (other_teller = 1; other_teller <= num_tellers; ++other_teller) {

        nj = list_size[other_teller] + list_size[num_tellers + other_teller];
        distance = abs(teller - other_teller);

    /* Check whether the customer at the end of queue other_teller qualifies
       for being the jockeying choice so far. */

    if (other_teller != teller && nj > ni + 1 && distance < min_distance) {

        /* The customer at the end of queue other_teller is our choice so
           far for the jockeying customer, so remember his queue number and
           its distance from the destination queue. */

        jumper       = other_teller;
        min_distance = distance;
    }
}

/* Check to see whether a jockeying customer was found. */

if (jumper > 0) {

    /* A jockeying customer was found, so remove him from his queue. */

    list_remove(LAST, jumper);
```

图 2.34 函数 jockey 的 C 代码，银行模型

```
        /* Check to see whether the teller of his new queue is busy. */

        if (list_size[num_tellers + teller] > 0)

            /* The teller of his new queue is busy, so place the customer at the
               end of this queue. */

            list_file(LAST, teller);

        else {

            /* The teller of his new queue is idle, so tally the jockeying
               customer's delay, make the teller busy, and start service. */

            sampst(sim_time - transfer[1], SAMPST_DELAYS);
            list_file(FIRST, num_tellers + teller);
            transfer[3] = teller;  /* Define before event_schedule. */
            event_schedule(sim_time + expon(mean_service, STREAM_SERVICE),
                           EVENT_DEPARTURE);
        }
    }
}
```

图 2.34 （续）

报告生成器的代码如图 2.35 所示，从一个循环开始，累加各队列的平均数，以得到队列的平均总数，如前面说明过的那样；然后将其与本模型中的不同出纳台数一起输出。最后，调用只有一个 sampst 变量的标准格式输出函数，输出队列中顾客延误时间的平均值和最大值。

```
void report(void)  /* Report generator function. */
{
    int   teller;
    float avg_num_in_queue;

    /* Compute and write out estimates of desired measures of performance. */

    avg_num_in_queue = 0.0;
    for (teller = 1; teller <= num_tellers; ++teller)
        avg_num_in_queue += filest(teller);
    fprintf(outfile, "\n\nWith%2d tellers, average number in queue = %10.3f",
            num_tellers, avg_num_in_queue);
    fprintf(outfile, "\n\nDelays in queue, in minutes:\n");
    out_sampst(outfile, SAMPST_DELAYS, SAMPST_DELAYS);
}
```

图 2.35　函数 report 的 C 代码，银行模型

2.6.3 仿真输出与讨论

图 2.36 所示的为仿真结果（在文件 mtbank.out 中）。与当前 5 个出纳台的策略相比，减少到 4 个出纳台看上去将在队列延误时间以及队列长度两方面都大大牺牲了顾客服务的质量。换一个方向，增加第六个出纳台将使顾客服务及平均队列长度得到本质改进；经济上是否可取，则取决于改善顾客服务带来的管理价值相对于增加一个出纳台的费用的情况如何。该例中增加第七个出纳台的必要性似乎不大，因为相对于 6 个出纳台系统，加上之后服务的改进并不大。还要注意，我们知道每个不同的系统一天当中服务的顾客有多少，即观测到的延误总数。该值在不同的系统中变化很小，因为到达速率相同，且假设大厅容量无限制。

```
Multiteller bank with separate queues & jockeying

Number of tellers                4 to  7

Mean interarrival time       1.000 minutes

Mean service time            4.500 minutes

Bank closes after            8.000 hours


With 4 tellers, average number in queue =      51.319

Delays in queue, in minutes:

 sampst                            Number
variable                             of
 number        Average             values          Maximum          Minimum
________________________________________________________________________________

    1          63.2229            501.000          156.363         0.000000
________________________________________________________________________________


With 5 tellers, average number in queue =       2.441

Delays in queue, in minutes:

 sampst                            Number
variable                             of
 number        Average             values          Maximum          Minimum
________________________________________________________________________________

    1          2.48149            483.000          21.8873         0.000000
________________________________________________________________________________


With 6 tellers, average number in queue =       0.718

Delays in queue, in minutes:

 sampst                            Number
variable                             of
 number        Average             values          Maximum          Minimum
________________________________________________________________________________

    1         0.763755            467.000          16.5103         0.000000
________________________________________________________________________________


With 7 tellers, average number in queue =       0.179

Delays in queue, in minutes:

 sampst                            Number
variable                             of
 number        Average             values          Maximum          Minimum
________________________________________________________________________________

    1         0.176180            493.000          6.97122         0.000000
________________________________________________________________________________
```

图 2.36　输出报告，银行模型

习题 2.4(b)和(c)细化了本模型，添加了新的输出度量(服务台利用率的度量，以及队列中顾客总数的最大值)；习题 2.4(d)进一步增强了该模型，它考虑了实际可能性，即银

行大厅容纳队列中的顾客的容量有限时的情形。

2.7 加工车间模型

本节我们用 Simlib 来仿真一个制造系统模型。本例是我们讨论的最复杂的例子，它说明仿真如何能用于发现一个生产过程中的瓶颈。

2.7.1 问题描述

制造系统由 5 个工作站组成，目前，工作站 1，2，…，5 分别有 3，2，4，3 和 1 台完全一样的机器，如图 2.37 所示。实际上，该系统是一个由 5 个多服务台队列的网络。假设工件到达系统的间隔时间为独立同分布(IID)指数随机变量，均值为 0.25 小时。一共有三类工件，记为类型 1、2 和 3，工件到达概率分别为 0.3、0.5 和 0.2。工件类型 1、2 和 3 分别需要 4、3 和 5 项任务才能完成，每项任务必须由指定的工作站按照规定的顺序完成。不同类型工件的工艺路线如表 2.18 所示。

表 2.18

工件类型	工艺路线中的工作站
1	3，1，2，5
2	4，1，3
3	2，5，1，4，3

图 2.37 有 5 个工作站的制造系统，表示了 1 类工件的工艺路线

因此，类型 2 工件的任务首先在工作站 4 做，然后在工作站 1 做，最后在工作站 3 做。

如果工件到达某一工作站，发现该工作站中所有机器都处于忙碌状态，则该工件进入该工作站的单一先进先出(FIFO)队列。在某一机器上完成任务的时间是一个独立的 2-厄兰型随机变量，其均值决定于工件类型，以及机器所属工作站(如果 X 是一个均值为 r 的 2-厄兰型随机变量，Y_1 和 Y_2 为独立的指数随机变量，每个均值为 $r/2$，那么 $X=Y_1+Y_2$，换句话说，X 称为伽马随机变量，形状参数为 2，比例参数为 $r/2$。详情参见第 6.2.2 小节)。之所以选择2-厄兰分布来表示服务时间，是因为经验表明，如果收集完成某个任务的时间数据，这些数据的直方图通常类似于厄兰分布的密度函数。各工件各项任务的平均服务时间如表 2.19 所示。

表 2.19

工件类型	任务顺次平均服务时间(单位：小时)
1	0.50，0.60，0.85，0.50
2	1.10，0.80，0.75
3	1.20，0.25，0.70，0.90，1.00

因此，类型 2 工件在工作站 4 需要的平均服务时间为 1.10 小时(即第一项任务完成)。

假设系统按天连续运行不失连续性，我们仿真系统 365 个八小时工作日，并估计每类工件在队列中的期望平均总延误时间(不含服务时间)，以及期望的全部平均工件总延误时间。我们使用真实的工件类型概率 0.3、0.5 和 0.2 来作为计算后者的权重。另外，还将估计队列中期望平均数，期望利用率(使用 Simlib 的函数 timest)，以及每个工作站队列中期望平均延误时间。

假设所有机器的成本近似相同，且系统着眼于改进效率可以购买新机器。我们将利用

上述仿真结果，将决定还需增加什么样的仿真运行（每一次新的仿真运行将包括 14 台机器，比原来的多 1 台）。在这些增加的仿真运行中，我们将使用全部平均工件总延误时间作为系统应购买何种类型机器的决策依据。

2.7.2 Simlib 程序

本模型的事件非常简明，如表 2.20 所示。

表 2.20

事件描述	事件类型
工件到达系统	1
工件从某一工作站离去	2
仿真结束	3

注意，对于本模型，离去事件是指工件离开其工艺路线上的任一工作站，而并不表示它离开系统，除非该离去是来自其工艺路线上最后一个工作站。该模型事件图如图 2.38 所示。

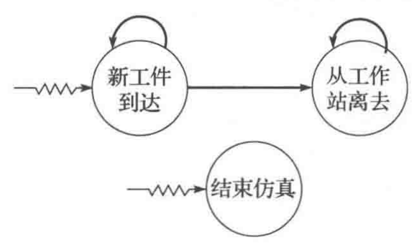


图 2.38 事件图，加工车间模型

我们使用表 2.21 所示的结构。

表 2.21

表	属性 1	属性 2	属性 3	属性 4
1～5：队列	到达工作站的时间	工件类型	任务编号	——
25：事件列表	事件时间	事件类型	工件类型	任务编号

队列表属性 1 的“到达时间”是指工件到达该表对应的工作站的时间，而不是最初到达系统的时间。工件的“任务编号”表示工件在其工艺路线上有多远，即第一项任务为 1，第二项任务为 2，以此类推；例如，类型 3 工件的任务编号 2 表示它在工作站 5 的加工过程。因此，工件的工作站可通过了解工件类型及任务编号来确定。

队列中工件的延误时间在本模型中用于不同途径，因此 sampst 变量结构比我们之前的模型更加丰富。我们需要各工作站队列中的平均延误时间（不管工件类型），则 sampst 变量 1～5 将用于此。此外，还需得到每类工件访问的所有队列中的平均延误时间（不管工作站），为此，将使用 sampst 变量 6～8，如表 2.22 所示。

表 2.22

sampst 变量编号	含义
1	工作站 1 队列中的延误时间
2	工作站 2 队列中的延误时间
3	工作站 3 队列中的延误时间
4	工作站 4 队列中的延误时间
5	工作站 5 队列中的延误时间
6	类型 1 工件在队列中的延误时间
7	类型 2 工件在队列中的延误时间
8	类型 3 工件在队列中的延误时间

因此，每个队列中的每个延误时间都将记录到两个不同的 sampst 变量，一个用于工作站，一个用于工件类型。

对于连续时间统计，我们将与以前一样使用 filest，现在同时还将使用 timest。由于每个队列都有一个表，因此我们可以通过使用 filest 易于得到每个队列中的时间平均数。我们还需观测每个工作站的利用率；由于一个工作站中可能有不止一台机器，因此利用率将定义为工作站中忙碌机器的时间平均数，再除以工作站中的机器总数。为了得到一个工作站中忙碌机器的平均数，我们将使用自定义(非 Simlib)数组 num_machines_busy[j]，我们将该数组保存当前在工作站 j 中忙碌的机器台数，并在一旦任意工作站改变该值时调用 timest。因此，我们有如表 2.23 所示的 timest 变量编号。

表 2.23

timest 变量编号	含义
1	工作站 1 中忙碌机器的个数
2	工作站 2 中忙碌机器的个数
3	工作站 3 中忙碌机器的个数
4	工作站 4 中忙碌机器的个数
5	工作站 5 中忙碌机器的个数

对于本模型，需要三类随机变量，对它们我们分别赋以表 2.24 所示的流。

表 2.24

流	用途
1	到达间隔时间
2	工件类型
3	服务时间

流 3 用来生成所有工件的服务时间，不管工件类型如何；在某些仿真中我们还可能需要用一个独立的流来生成每类工件或在每个工作站的服务时间，以便控制每类工件或每个工作站的具体特性。

该程序的外部定义如图 2.39 所示。在必需的“#include simlib.h”之后，我们定义符号常量以用于事件类型、随机数流号，以及工作站个数与工件类型的最大值；这些最大值将用于分配数组的空间，这样做而不是在数组声明中直接规定其数值，使得程序更具通用性。接下来我们声明几个整型变量及整型数组，其名称都非常不言自明；我们使用 i 作为工件类型的索引号，j 作为工作站或者任务编号索引。工作站 j 中的机器个数为 num_machines[j]，类型 i 工件的任务总数(工作站访问次数)为 num_tasks[i]，route[i][j]表示类型 i 工件的任务 j 所在工作站。浮点变量及浮点数组声明如下：mean_interarrival 的单位为小时(本模型的时间单位)，但仿真长度 length_simulation 的单位为 8 小时的天数(=365)，因此必须进行单位换算；prob_distrib_job_type[i]为工件类型小于或等于 i 的概率；mean_service[i][j]为类型 i 工件的任务 j 的平均服务时间(单位：小时)。再接下来是函数声明；注意，整型参数 new_job 传送给函数 arrive，其中值为 1 表示其为系统的一个新的到达(在这种情况下，arrive 作为一个事件函数提供服务)；值为 2 表示其为非事件，即工件离开一个工作站并按工艺路线“到达”下一个工作站(在这种情况下，arrive 作为非事件“效用”函数)。

主函数如图 2.40 所示，有点长但仍是通常形式。注意，为保持时间单位的一致性，正确的做法是调用 event_schedule 来安排结束仿真事件。

```
/* External definitions for job-shop model. */

#include "simlib.h"              /* Required for use of simlib.c. */

#define EVENT_ARRIVAL         1  /* Event type for arrival of a job to the
                                    system. */
#define EVENT_DEPARTURE       2  /* Event type for departure of a job from a
                                    particular station. */
#define EVENT_END_SIMULATION  3  /* Event type for end of the simulation. */
#define STREAM_INTERARRIVAL   1  /* Random-number stream for interarrivals. */
#define STREAM_JOB_TYPE       2  /* Random-number stream for job types. */
#define STREAM_SERVICE        3  /* Random-number stream for service times. */
#define MAX_NUM_STATIONS      5  /* Maximum number of stations. */
#define MAX_NUM_JOB_TYPES     3  /* Maximum number of job types. */

/* Declare non-simlib global variables. */

int   num_stations, num_job_types, i, j, num_machines[MAX_NUM_STATIONS + 1],
      num_tasks[MAX_NUM_JOB_TYPES +1],
      route[MAX_NUM_JOB_TYPES +1][MAX_NUM_STATIONS + 1],
      num_machines_busy[MAX_NUM_STATIONS + 1], job_type, task;
float mean_interarrival, length_simulation, prob_distrib_job_type[26],
      mean_service[MAX_NUM_JOB_TYPES +1][ MAX_NUM_STATIONS + 1];
FILE  *infile, *outfile;

/* Declare non-simlib functions. */

void  arrive(int new_job);
void  depart(void);
void  report(void);
```

图 2.39　外部定义的 C 代码，加工车间模型

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("jobshop.in",  "r");
    outfile = fopen("jobshop.out", "w");

    /* Read input parameters. */

    fscanf(infile, "%d %d %f %f", &num_stations, &num_job_types,
           &mean_interarrival, &length_simulation);
    for (j = 1; j <= num_stations; ++j)
        fscanf(infile, "%d", &num_machines[j]);
    for (i = 1; i <= num_job_types; ++i)
        fscanf(infile, "%d", &num_tasks[i]);
    for (i = 1; i <= num_job_types; ++i) {
        for (j = 1; j <= num_tasks[i]; ++j)
            fscanf(infile, "%d", &route[i][j]);
        for (j = 1; j <= num_tasks[i]; ++j)
            fscanf(infile, "%f", &mean_service[i][j]);
    }
    for (i = 1; i <= num_job_types; ++i)
        fscanf(infile, "%f", &prob_distrib_job_type[i]);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Job-shop model\n\n");
    fprintf(outfile, "Number of workstations%21d\n\n", num_stations);
    fprintf(outfile, "Number of machines in each station     ");
    for (j = 1; j <= num_stations; ++j)
        fprintf(outfile, "%5d", num_machines[j]);
```

图 2.40　主函数的 C 代码，加工车间模型

```
    fprintf(outfile, "\n\nNumber of job types%25d\n\n", num_job_types);
    fprintf(outfile, "Number of tasks for each job type      ");
    for (i = 1; i <= num_job_types; ++i)
        fprintf(outfile, "%5d", num_tasks[i]);
    fprintf(outfile, "\n\nDistribution function of job types  ");
    for (i = 1; i <= num_job_types; ++i)
        fprintf(outfile, "%8.3f", prob_distrib_job_type[i]);
    fprintf(outfile, "\n\nMean interarrival time of jobs%14.2f hours\n\n",
            mean_interarrival);
    fprintf(outfile, "Length of the simulation%20.1f eight-hour days\n\n\n",
            length_simulation);
    fprintf(outfile, "Job type     Workstations on route");
    for (i = 1; i <= num_job_types; ++i) {
        fprintf(outfile, "\n\n%4d        ", i);
        for (j = 1; j <= num_tasks[i]; ++j)
            fprintf(outfile, "%5d", route[i][j]);
    }
    fprintf(outfile, "\n\n\nJob type     ");
    fprintf(outfile, "Mean service time (in hours) for successive tasks");
    for (i = 1; i <= num_job_types; ++i) {
        fprintf(outfile, "\n\n%4d    ", i);
        for (j = 1; j <= num_tasks[i]; ++j)
            fprintf(outfile, "%9.2f", mean_service[i][j]);
    }

    /* Initialize all machines in all stations to the idle state. */

    for (j = 1; j <= num_stations; ++j)
        num_machines_busy[j] = 0;

    /* Initialize simlib */

    init_simlib();

    /* Set maxatr = max(maximum number of attributes per record, 4) */

    maxatr = 4;  /* NEVER SET maxatr TO BE SMALLER THAN 4. */

    /* Schedule the arrival of the first job. */

    event_schedule(expon(mean_interarrival, STREAM_INTERARRIVAL),
                   EVENT_ARRIVAL);

    /* Schedule the end of the simulation.  (This is needed for consistency of
       units.) */

    event_schedule(8 * length_simulation, EVENT_END_SIMULATION);

    /* Run the simulation until it terminates after an end-simulation event
       (type EVENT_END_SIMULATION) occurs. */

    do {

        /* Determine the next event. */

        timing();

        /* Invoke the appropriate event function. */

        switch (next_event_type) {
            case EVENT_ARRIVAL:
                arrive(1);
                break;
            case EVENT_DEPARTURE:
                depart();
```

图 2.40 （续）

```
            break;
        case EVENT_END_SIMULATION:
            report();
            break;
    }

/* If the event just executed was not the end-simulation event (type
   EVENT_END_SIMULATION), continue simulating.  Otherwise, end the
   simulation. */

} while (next_event_type != EVENT_END_SIMULATION);

fclose(infile);
fclose(outfile);

return 0;
}
```

图 2.40 （续）

函数 arrive 的流程图和代码清单分别如图 2.41 和图 2.42 所示。它一开始检查 new_job 来决定，要么它是用作事件函数来处理系统的一个新到达工件(new_job=1)，要么它

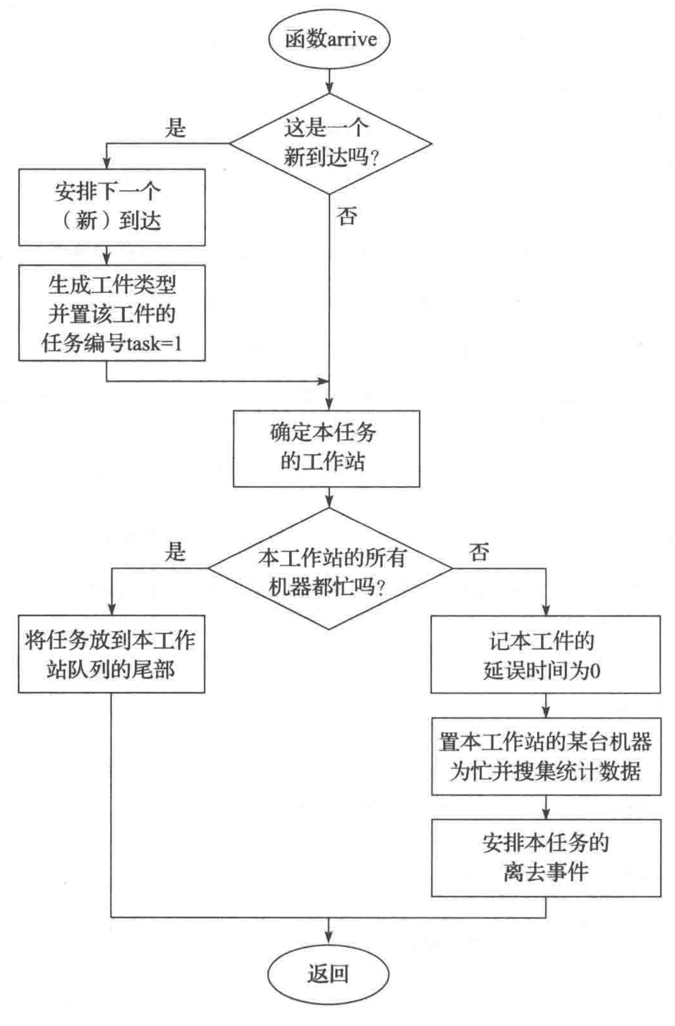


图 2.41 到达函数的流程图，加工车间模型

用作离开工作站事件的最后一部分，以处理现有工件按其工艺路线中到达下一工作站(new_job=2)。如果这是一个新到达工件，则安排下一到达；新到达工件的类型为在整数 1 至 3 之间的随机数，这利用 Simlib 函数 random_integer 以及 prob_distrib_job_type 中的累积概率来产生；最后，将这个新工件的任务编号初始化为 1。如下面我们在讨论 depart 时将要看到的那样，如果这不是一个新的到达，则工件类型和任务编号都已经在全局变量 job_type 和 task 中正确地给出了值。不管工件是否是新的，通过查询工艺路线数组，函数将继续根据工件类型及任务编号来确定该到达的工作站。接下来，检查该工作站中的所有机器是否都在忙碌。如果是，则只需将工件置于该站队尾。如果不是，则工件在此处的延误时间为零(该工作站和工件类型都要在 sampst 中加以记录)，将该站的某个机器置成忙碌，且在在一个合适的 timest 变量中加以标记。注意，按照 timest 的要求，须将整数型数组 num_machines_busy 转换成浮点型。最后，安排该工件退出此工作站，注意，在调用 event_schedule 之前要定义事件记录前两个属性之外的属性。

```
void arrive(int new_job)  /* Function to serve as both an arrival event of a job
                             to the system, as well as the non-event of a job's
                             arriving to a subsequent station along its
                             route. */
{
    int station;

    /* If this is a new arrival to the system, generate the time of the next
       arrival and determine the job type and task number of the arriving
       job. */


    if (new_job == 1) {

    event_schedule(sim_time + expon(mean_interarrival, STREAM_INTERARRIVAL),
                   EVENT_ARRIVAL);
    job_type = random_integer(prob_distrib_job_type, STREAM_JOB_TYPE);
    task     = 1;
}

/* Determine the station from the route matrix. */

station = route[job_type][task];

/* Check to see whether all machines in this station are busy. */

if (num_machines_busy[station] == num_machines[station]) {

    /* All machines in this station are busy, so place the arriving job at
       the end of the appropriate queue. Note that the following data are
       stored in the record for each job:
         1. Time of arrival to this station.
         2. Job type.
         3. Current task number. */

    transfer[1] = sim_time;
    transfer[2] = job_type;
    transfer[3] = task;
    list_file(LAST, station);
}

else {

    /* A machine in this station is idle, so start service on the arriving
       job (which has a delay of zero). */

    sampst(0.0, station);                                  /* For station. */
```

图 2.42 函数 arrive 的 C 代码，加工车间模型

```
        sampst(0.0, num_stations + job_type);              /* For job type. */
        ++num_machines_busy[station];
        timest((float) num_machines_busy[station], station);

        /* Schedule a service completion.  Note defining attributes beyond the
           first two for the event record before invoking event_schedule. */

        transfer[3] = job_type;
        transfer[4] = task;
        event_schedule(sim_time
                       + erlang(2, mean_service[job_type][task],
                                STREAM_SERVICE),
                       EVENT_DEPARTURE);
    }
}
```

图 2.42 （续）

事件函数 depart 的流程图及代码清单如图 2.43 和图 2.44 所示。离去工件的 job_type 和 task 的值由离去事件记录以及工作站“station”获得，离去事件记录刚刚由定时函数放在 transfer 数组中，工作站“station”是本工件正在离开，然后在工艺路线数组中查找的工作站。如果该站队列为空，则该站内的一台机器变为空闲状态，同时记录于 timest 中。如果有排队的，队首工件从队列删除（工件类型为 job_type_queue，任务编号为 task_

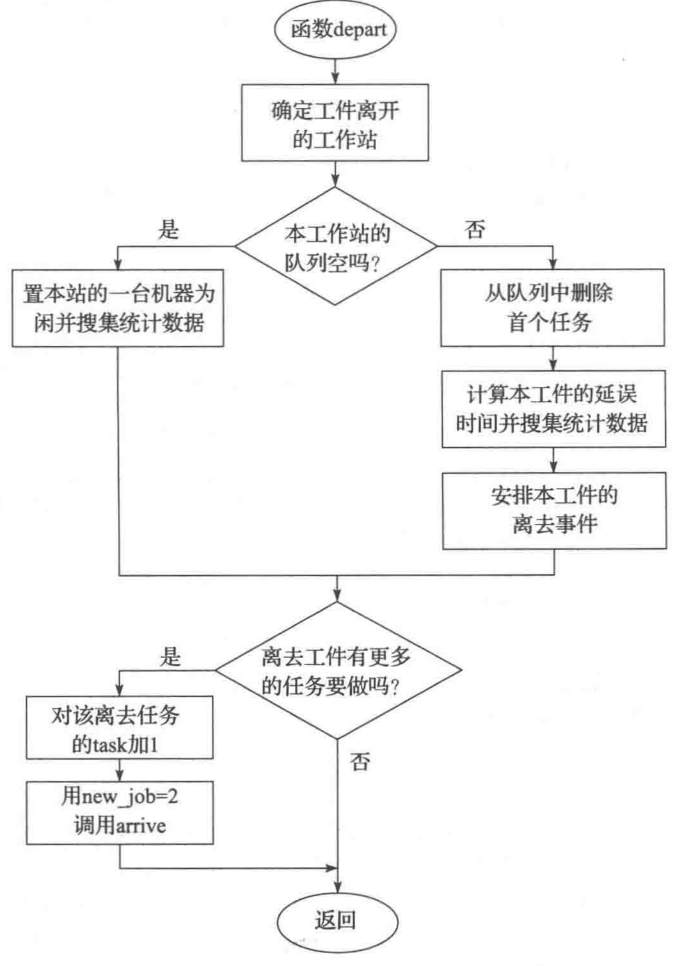


图 2.43 离去函数的流程图，加工车间模型

queue，区别于之前离开该站的那个工件)，其延误存于 sampst 的两个相应变量中，并安排离开该站；还有，对于长仿真，sim_time 和 transfer 都必须设为 double 型，以防计算延误时间做减法时过多的四舍五入错误。最后如果离开该站的工件仍有更多的任务要做，则任务编号加 1，并调用 arrive 将其送到其工艺路线中的下一工作站，其中参数 new_job 值为 2，说明这不是一个新到达的工件。

```
void depart(void)  /* Event function for departure of a job from a particular
                      station. */
{
    int station, job_type_queue, task_queue;

    /* Determine the station from which the job is departing. */

    job_type = transfer[3];
    task     = transfer[4];
    station  = route[job_type][task];

    /* Check to see whether the queue for this station is empty. */

    if (list_size[station] == 0) {

        /* The queue for this station is empty, so make a machine in this
           station idle. */

        --num_machines_busy[station];
        timest((float) num_machines_busy[station], station);
    }

    else {

        /* The queue is nonempty, so start service on first job in queue. */

        list_remove(FIRST, station);

        /* Tally this delay for this station. */

        sampst(sim_time - transfer[1], station);

        /* Tally this same delay for this job type. */

        job_type_queue = transfer[2];
        task_queue     = transfer[3];
        sampst(sim_time - transfer[1], num_stations + job_type_queue);

        /* Schedule end of service for this job at this station.  Note defining
           attributes beyond the first two for the event record before invoking
           event_schedule. */

        transfer[3] = job_type_queue;
        transfer[4] = task_queue;
        event_schedule(sim_time
                       + erlang(2, mean_service[job_type_queue][task_queue],
                                STREAM_SERVICE),
                       EVENT_DEPARTURE);
}

    /* If the current departing job has one or more tasks yet to be done, send
       the job to the next station on its route. */

    if (task < num_tasks[job_type]) {
        ++task;
        arrive(2);
    }
}
```

图 2.44　函数 depart 的 C 代码，加工车间模型

报告生成函数的代码如图 2.45 所示。第一个 for 循环用于计算每一种类型 i 工件在所有队列中平均总延误时间；这里“总”字指的是每类工件在其工艺路线上所有队列的平均延误时间求和。我们必须将该类工件的 sampst 返回的平均值乘以任务数 num_tasks[i]，因为 sampst 被该类工件的每项任务都调用过，每项任务离开系统 num_tasks[i]次而不是一次，从而用 sampst 计算均值时使得分母扩大了 num_tasks[i]倍。然后对这些平均总延误时间进行加权，权值是各工件类型的概率，再加起来得到整体平均总延误时间；相比于忽略工件类型只是简单地对所有工件总延误时间求平均，用这些工件类型的真实(精确)概率所获得的估计更加精确(可变性较小)。此外，须取 prob_distrib_job_type 数组中连续差来代替工件类型的概率，因为这个数组是累加概率(技术提示：以上 sampst 平均乘以 num_tasks[i]稍微有些不准确。由于仿真结束时通常仍有些工件留在系统中，还未完成在所有队列中的延误，因此它们不应该有任何延误记录在 sampst 中。然而，由于本仿真的长度共 365×8 小时＝2 920小时，且每小时平均期望到达 4 个工件，将期望有 11 680 个工件到达，因此这点误差多半是很小的。另一种收集各类工件队列中总延误时间的方法，可避免上述问题，参见习题 2.5)。该函数以 for 循环结束，对每个工作站 j，输出队列中时间平均数、利用率(用忙碌机器的时间平均数除以工作站中的机器数计算得出)，以及队列中平均延误时间。

```
void report(void)  /* Report generator function. */
{
    int   i;
    float overall_avg_job_tot_delay, avg_job_tot_delay, sum_probs;

    /* Compute the average total delay in queue for each job type and the
       overall average job total delay. */

    fprintf(outfile, "\n\n\n\nJob type     Average total delay in queue");
    overall_avg_job_tot_delay = 0.0;
    sum_probs                 = 0.0;
    for (i = 1; i <= num_job_types; ++i) {
        avg_job_tot_delay = sampst(0.0, -(num_stations + i)) * num_tasks[i];
        fprintf(outfile, "\n\n%4d%27.3f", i, avg_job_tot_delay);
        overall_avg_job_tot_delay += (prob_distrib_job_type[i] - sum_probs)
                                     * avg_job_tot_delay;
        sum_probs = prob_distrib_job_type[i];
    }
    fprintf(outfile, "\n\nOverall average job total delay =%10.3f\n",
            overall_avg_job_tot_delay);

    /* Compute the average number in queue, the average utilization, and the
       average delay in queue for each station. */

    fprintf(outfile,
           "\n\n\n Work      Average number      Average       Average delay");
    fprintf(outfile,
             "\nstation        in queue       utilization        in queue");
    for (j = 1; j <= num_stations; ++j)
        fprintf(outfile, "\n\n%4d%17.3f%17.3f%17.3f", j, filest(j),
                timest(0.0, -j) / num_machines[j], sampst(0.0, -j));
}
```

图 2.45　函数 report 的 C 代码，加工车间模型

产生 m-厄兰随机变量的函数在附录 2A 的图 2.66 中(它是 Simlib 的一部分，而非本模型的)，与前面所述厄兰分布的物理模型一致。注意，我们必须将要求的最终厄兰随机变量的期望除以 m，以得到其组分指数型随机变量的期望。另外，用户自定义流编号 stream，在此作为输入并只要传给指数变量发生器(见附录 2A 图 2.63)，然后传给随机数生成器 lcgrand。

2.7.3 仿真输出与讨论

图 2.46 显示了本节仿真的输出文件(jobshop. out)。通过工件类型加权，工件在队列中等待所花费的平均时间将近 11 小时；这并不是在系统中的平均时间，因为它并不包括在工作站的处理时间(参见习题 2.6)。我们需要补充一点，该模型在不同计算机系统上可能产生不同的数值结果，这是由于其长度和复杂度不同，产生不同舍入误差，以及使用随机数流的顺序的改变的缘故。

```
Job-shop model

Number of workstations                        5

Number of machines in each station             3    2    4    3    1

Number of job types                            3

Number of tasks for each job type              4    3    5

Distribution function of job types         0.300   0.800   1.000

Mean interarrival time of jobs              0.25 hours

Length of the simulation                   365.0 eight-hour days


Job type     Workstations on route

   1            3    1    2    5

   2            4    1    3

   3            2    5    1    4    3


Job type     Mean service time (in hours) for successive tasks

   1         0.50      0.60      0.85      0.50

   2         1.10      0.80      0.75

   3         1.20      0.25      0.70      0.90      1.00


Job type     Average total delay in queue

   1                     10.022

   2                      9.403

   3                     15.808

Overall average job total delay =     10.870


 Work       Average number       Average        Average delay
station        in queue        utilization        in queue

   1           12.310             0.969            3.055

   2           11.404             0.978            5.677

   3            0.711             0.719            0.177

   4           17.098             0.961            6.110

   5            2.095             0.797            1.043
```

图　2.46

观察工作站的统计可知，工作站 1、2 和 4 出现了瓶颈，尽管其严重性的顺序取决于我们观察的是队列中平均数、利用率，还是队列平均延误时间的结果。因此，我们对这三个工作站每个逐次增加一台机器，增加了三次仿真运行(工作站 3 和 5 看上去相对不那么拥挤，因此我们不考虑给它们增加新机器)，以观察哪类新机器将给系统效率带来最大影响。采用所有平均工件总延误时间作为单一的性能度量，这些附加的仿真的结果如表 2.25 所示。如果只是简单观察这些数值，我们可以看到机器显然应该加到工作站 4 上，使得所有平均工件总延误时间减少最多。然而如通常的情况那样，这个结论只是试验性的，因为我们对每个模型变量只运行了一次仿真而已；在本例中尤为突出，因为三种新机器的配置的结果实在是太接近了。

表 2.25　当前及建议的机器配置的估计期望整个平均工件总延误时间

工作站中的机器数	整个平均工件总延误时间(单位：小时)
3，2，4，3，1(当前配置)	10.9
4，2，4，3，1(工作站 1 增加一台机器)	8.1
3，3，4，3，1(工作站 2 增加一台机器)	7.6
3，2，4，4，1(工作站 4 增加一台机器)	7.5

2.8　高效的事件表处理

本章和前一章讨论的动态仿真的共同点是，都需要按照某种方式调度将来事件，并决定这些已调度的事件哪个即将发生。我们已经看到了两种不同的处理事件表的方法。在第 1 章中，表是按顺序存储的，存储索引号就是事件类型，下一发生的事件是根据从上到下搜索事件表以得到最小事件时间而确定的。而在本章中，借助于 Simlib 处理链表的能力，我们将事件表存储为双链表，按照事件时间属性递增排序，并将另一属性用于事件类型；因为它总在事件列表顶端，则易于确定下一事件。然而，向事件表中添加一个事件却需要更多工作，因为它要搜索其正确的位置。不论哪种方式，无论在取走一个事件，还是在存入一个事件，都需要搜索事件表。

动态仿真中对某类事件表处理的需求使得大量学者研究是否有其他更快的方法，至少对某些类仿真来说如此。对涉及众多事件的复杂仿真来说，执行仿真所需的许多计算机时间都花费在事件表的处理上。Comfort(1981)报告说，在某一类样例模型中，处理事件表所需的指令数甚至可达到整个仿真指令总数的 40%。McCormack 和 Sargent(1981)给出了另外的证据显示，事件处理算法的选择对仿真执行时间有重大影响。Henriksen(1983)用“惊人的失败”一词来描述简单的自顶向下搜索将事件记录插入事件表的方法的性能(在一个样例模型的情况下)。

一个改进 Simlib 搜索新事件正确位置的方法是使用更有效的搜索技术。一个著名的搜索算法，称为二叉树搜索，引入事件表的另一个指针(除头尾指针之外)，该指针总是指向事件列表的中间记录，或两个中间记录之中的一个(若记录数为偶数)。当把一个新的事件记录放到表中时，首先检查中间指针所指向(或相邻)的记录，来决定新记录是应该置于事件表的前半部分还是后半部分。然后顺序搜索合适的那一半表以决定新记录的位置(严格来讲，术语“二叉树搜索”意味着选出的那一半表本身应再分成两个四分之一表，合适的部分再分，以此类推。然而，如果事件表是以链表存储的，那么这样做将导致需维护许多指针来区分所有这些分隔点)。对于事件表可能很长的仿真(例如 2.5 节中终端数很大的的计算机分时模型就是这样)，这种方法对整个计算时间会产生实实在在的差别(参见习题 2.34)。

还有很多其他算法用于事件表处理，包括各种树和堆[参见 Knuth(1998b)]，日历队列[Brown(1988)]，梯形对列[Tang 等(2005)]，以及亨利克森(henriksen's algorithm)

(1977)。有些文章给出了各种事件列表算法的经验评估，如 chuang 等(1973)、Jones (1986)和 Ronngren 和 Ayani(1997)。

最好的事件表处理算法的选择可能取决于仿真的类型、用到的参数和分布，以及那些影响事件在事件表中分布的其他因素。例如有这样一种仿真，事件安排(即放入事件表中)与事件发生(即从事件表中取出)之间的时间跨度对所有事件都差不多相同，那么我们更倾向于直接将事件插入链表数据结构的表尾部；在这种情况下，自下而上搜索事件表也许会更好，因为大部分情形搜索将很快结束。大多数现代仿真软件包(见第 3 章)使用高效事件表处理算法。

附录 2A　Simlib 的 C 代码

Simlib 函数的 C 代码如图 2.47 至图 2.66 所示。头文件 simlib.h 在图 2.47 中给出，用户必须 #include 它；然后该文件 #include 如图 2.48，图 2.49 至图 2.66 所示的各个 simlibdefs.h，加上附录 7A 图 7.5 中的随机数发生器 lcgrand，组成了文件 simlib.c。所有这些代码可从 www.mhhe.com/law 下载。

在图 2.57 所示的 timest 中，仿真钟 sim_time 与上一事件时间 tlvc[]两者之间要做减法。对于长仿真，相对于它们的差值，这两个值可能非常大，因此需要使用 double 类型以避免该减法中的精度损失。

```
/* This is simlib.h. */

/* Include files. */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "simlibdefs.h"
/* Declare simlib global variables. */

extern int    *list_rank, *list_size, next_event_type, maxatr, maxlist;
extern float  *transfer, sim_time, prob_distrib[26];
extern struct master {
    float  *value;
    struct master *pr;
    struct master *sr;
} **head, **tail;

/* Declare simlib functions. */

extern void  init_simlib(void);
extern void  list_file(int option, int list);
extern void  list_remove(int option, int list);
extern void  timing(void);
extern void  event_schedule(float time_of_event, int type_of_event);
extern int   event_cancel(int event_type);
extern float sampst(float value, int varibl);
extern float timest(float value, int varibl);
extern float filest(int list);
extern void  out_sampst(FILE *unit, int lowvar, int highvar);
extern void  out_timest(FILE *unit, int lowvar, int highvar);
extern void  out_filest(FILE *unit, int lowlist, int highlist);
extern float expon(float mean, int stream);
extern int   random_integer(float prob_distrib[], int stream);
extern float uniform(float a, float b, int stream);
extern float erlang(int m, float mean, int stream);
extern float lcgrand(int stream);
extern void  lcgrandst(long zset, int stream);
extern long  lcgrandgt(int stream);
```

图 2.47　头文件 simlib.h

```
/* This is simlibdefs.h. */

/* Define limits. */

#define MAX_LIST    25      /* Max number of lists. */
#define MAX_ATTR    10      /* Max number of attributes. */
#define MAX_SVAR    25      /* Max number of sampst variables. */
#define TIM_VAR     25      /* Max number of timest variables. */
#define MAX_TVAR    50      /* Max number of timest variables + lists. */
#define EPSILON      0.001  /* Used in event_cancel. */

/* Define array sizes. */

#define LIST_SIZE   26      /* MAX_LIST + 1. */
#define ATTR_SIZE   11      /* MAX_ATTR + 1. */
#define SVAR_SIZE   26      /* MAX_SVAR + 1. */
#define TVAR_SIZE   51      /* MAX_TVAR + 1. */

/* Define options for list_file and list_remove. */

#define FIRST        1      /* Insert at (remove from) head of list. */
#define LAST         2      /* Insert at (remove from) end of list. */
#define INCREASING   3      /* Insert in increasing order. */
#define DECREASING   4      /* Insert in decreasing order. */

/* Define some other values. */

#define LIST_EVENT  25      /* Event list number. */
#define INFINITY     1.E30  /* Not really infinity, but a very large number. */

/* Pre-define attribute numbers of transfer for event list. */

#define EVENT_TIME   1      /* Attribute 1 in event list is event time. */
#define EVENT_TYPE   2      /* Attribute 2 in event list is event type. */
```

图 2.48　包括的文件 simlibdefs.h

```
/* This is simlib.c (adapted from SUPERSIMLIB, written by Gregory Glockner). */

/* Include files. */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "simlibdefs.h"

/* Declare simlib global variables. */

int    *list_rank, *list_size, next_event_type, maxatr = 0, maxlist = 0;
float  *transfer, sim_time, prob_distrib[26];
struct master {
    float  *value;
    struct master *pr;
    struct master *sr;
} **head, **tail;

/* Declare simlib functions. */

void  init_simlib(void);
void  list_file(int option, int list);
void  list_remove(int option, int list);
void  timing(void);
void  event_schedule(float time_of_event, int type_of_event);
int   event_cancel(int event_type);
float sampst(float value, int variable);
float timest(float value, int variable);
float filest(int list);
void  out_sampst(FILE *unit, int lowvar, int highvar);
```

图 2.49　外部的 Simlib 定义

```
/* This is simlib.c (adapted from SUPERSIMLIB, written by Gregory Glockner). */

/* Include files. */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "simlibdefs.h"

/* Declare simlib global variables. */

int    *list_rank, *list_size, next_event_type, maxatr = 0, maxlist = 0;
float  *transfer, sim_time, prob_distrib[26];
struct master {
    float  *value;
    struct master *pr;
    struct master *sr;
} **head, **tail;

/* Declare simlib functions. */

void  init_simlib(void);
void  list_file(int option, int list);
void  list_remove(int option, int list);
void  timing(void);
void  event_schedule(float time_of_event, int type_of_event);
int   event_cancel(int event_type);
float sampst(float value, int variable);
float timest(float value, int variable);
float filest(int list);
void  out_sampst(FILE *unit, int lowvar, int highvar);
```

图 2.49 （续）

```
void init_simlib()
{

/* Initialize simlib.c.  List LIST_EVENT is reserved for event list, ordered by
   event time.  init_simlib must be called from main by user. */

    int list, listsize;

    if (maxlist < 1) maxlist = MAX_LIST;
    listsize = maxlist + 1;

    /* Initialize system attributes. */

    sim_time = 0.0;
    if (maxatr < 4) maxatr = MAX_ATTR;

    /* Allocate space for the lists. */

    list_rank = (int *)             calloc(listsize,   sizeof(int));
    list_size = (int *)             calloc(listsize,   sizeof(int));
    head      = (struct master **) calloc(listsize,   sizeof(struct master *));
    tail      = (struct master **) calloc(listsize,   sizeof(struct master *));
    transfer  = (float *)           calloc(maxatr + 1, sizeof(float));

    /* Initialize list attributes. */

    for(list = 1; list <= maxlist; ++list) {
        head [list]     = NULL;
        tail [list]     = NULL;
        list_size[list] = 0;
        list_rank[list] = 0;
    }
```

图 2.50　Simlib 函数 init_simlib

```
    /* Set event list to be ordered by event time. */

    list_rank[LIST_EVENT] = EVENT_TIME;

    /* Initialize statistical routines. */

    sampst(0.0, 0);
    timest(0.0, 0);
}
```

图 2.50 (续)

```
void list_file(int option, int list)
{

/* Place transfr into list "list".
   Update timest statistics for the list.
   option = FIRST place at start of list
            LAST  place at end of list
            INCREASING  place in increasing order on attribute list_rank(list)
            DECREASING  place in decreasing order on attribute list_rank(list)
            (ties resolved by FIFO) */

    struct master *row, *ahead, *behind, *ihead, *itail;
    int    item, postest;

    /* If the list value is improper, stop the simulation. */

    if(!((list >= 0) && (list <= MAX_LIST))) {
        printf("\nInvalid list %d for list_file at time %f\n", list, sim_time);
        exit(1);
    }

    /* Increment the list size. */

    list_size[list]++;

    /* If the option value is improper, stop the simulation. */

    if(!((option >= 1) && (option <= DECREASING))) {
        printf(
            "\n%d is an invalid option for list_file on list %d at time %f\n",
            option, list, sim_time);
        exit(1);
    }

    /* If this is the first record in this list, just make space for it. */

    if(list_size[list] == 1) {

        row        = (struct master *) malloc(sizeof(struct master));
        head[list] = row ;
        tail[list] = row ;
        (*row).pr  = NULL;
        (*row).sr  = NULL;
    }

    else { /* There are other records in the list. */

        /* Check the value of option. */

        if ((option == INCREASING) || (option == DECREASING)) {
            item = list_rank[list];
            if(!((item >= 1) && (item <= maxatr))) {
```

图 2.51 Simlib 函数 list_file

```
            printf(
                "%d is an improper value for rank of list %d at time %f\n",
                item, list, sim_time) ;
            exit(1);
        }

        row    = head[list];
        behind = NULL; /* Dummy value for the first iteration. */

        /* Search for the correct location. */

        if (option == INCREASING) {
            postest = (transfer[item] >= (*row).value[item]);
            while (postest) {
                behind  = row;
                row     = (*row).sr;
                postest = (behind != tail[list]);
                if (postest)
                    postest = (transfer[item] >= (*row).value[item]);
            }
        }

        else {

            postest = (transfer[item] <= (*row).value[item]);
            while (postest) {
                behind  = row;
                row     = (*row).sr;
                postest = (behind != tail[list]);
                if (postest)
                    postest = (transfer[item] <= (*row).value[item]);
            }
        }

        /* Check to see if position is first or last.  If so, take care of
           it below. */

        if (row == head[list])

            option = FIRST;

        else

            if (behind == tail[list])

                option = LAST;

            else { /* Insert between preceding and succeeding records. */

                ahead          = (*behind).sr;
                row            = (struct master *)
                                     malloc(sizeof(struct master));
                (*row).pr      = behind;
                (*behind).sr   = row;
                (*ahead).pr    = row;
                (*row).sr      = ahead;
            }
    } /* End if inserting in increasing or decreasing order. */

        if (option == FIRST) {
            row          = (struct master *) malloc(sizeof(struct master));
            ihead        = head[list];
            (*ihead).pr  = row;
            (*row).sr    = ihead;
            (*row).pr    = NULL;
            head[list]   = row;
        }
```

图 2.51 （续）

```
        if (option == LAST) {
            row          = (struct master *) malloc(sizeof(struct master));
            itail        = tail[list];
            (*row).pr    = itail;
            (*itail).sr = row;
            (*row).sr    = NULL;
            tail[list]   = row;
        }
    }

    /* Copy the row values from the transfer array. */

    (*row).value = (float *) calloc(maxatr + 1, sizeof(float));
    for (item = 0; item <= maxatr; ++item)
        (*row).value[item] = transfer[item];


    /* Update the area under the number-in-list curve. */

    timest((float)list_size[list], TIM_VAR + list);
}
```

图 2.51 （续）

```
void list_remove(int option, int list)
{

/* Remove a record from list "list" and copy attributes into transfer.
   Update timest statistics for the list.
   option = FIRST remove first record in the list
            LAST  remove last record in the list */

    struct master *row, *ihead, *itail;

    /* If the list value is improper, stop the simulation. */

    if(!((list >= 0) && (list <= MAX_LIST))) {
        printf("\nInvalid list %d for list_remove at time %f\n",
               list, sim_time);
        exit(1);
    }
    /* If the list is empty, stop the simulation. */

    if(list_size[list] <= 0) {
        printf("\nUnderflow of list %d at time %f\n", list, sim_time);
        exit(1);
    }

    /* Decrement the list size. */

    list_size[list]--;

    /* If the option value is improper, stop the simulation. */

    if(!(option == FIRST || option == LAST)) {
        printf(

        "\n%d is an invalid option for list_remove on list %d at time %f\n",
        option, list, sim_time);
    exit(1);
}

if(list_size[list] == 0) {
```

图 2.52 Simlib 函数 list_remove

```
        /* There is only 1 record, so remove it. */

        row        = head[list];
        head[list] = NULL;
        tail[list] = NULL;
    }

    else {

        /* There is more than 1 record, so remove according to the desired
           option. */

        switch(option) {

            /* Remove the first record in the list. */

            case FIRST:
                row          = head[list];
                ihead        = (*row).sr;
                (*ihead).pr  = NULL;
                head[list]   = ihead;
                break;

            /* Remove the last record in the list. */

            case LAST:
                row          = tail[list];
                itail        = (*row).pr;
                (*itail).sr  = NULL;
                tail[list]   = itail;
                break;
        }
    }

    /* Copy the data and free memory. */

    free((char *)transfer);
    transfer = (*row).value;
    free((char *)row);

    /* Update the area under the number-in-list curve. */

    timest((float)list_size[list], TIM_VAR + list);
}
```

图 2.52　(续)

```
void timing()
{

/* Remove next event from event list, placing its attributes in transfer.
   Set sim_time (simulation time) to event time, transfer[1].
   Set next_event_type to this event type, transfer[2]. */

    /* Remove the first event from the event list and put it in transfer[]. */

    list_remove(FIRST, LIST_EVENT);

    /* Check for a time reversal. */
```

图 2.53　Simlib 函数 timing

```
    if(transfer[EVENT_TIME] < sim_time) {
        printf(
            "\nAttempt to schedule event type %f for time %f at time %f\n",
            transfer[EVENT_TYPE], transfer[EVENT_TIME], sim_time);
        exit(1);
    }

    /* Advance the simulation clock and set the next event type. */

    sim_time        = transfer[EVENT_TIME];
    next_event_type = transfer[EVENT_TYPE];
}
```

图 2.53 (续)

```
void event_schedule(float time_of_event, int type_of_event)
{

/* Schedule an event at time event_time of type event_type.  If attributes
   beyond the first two (reserved for the event time and the event type) are
   being used in the event list, it is the user's responsibility to place their
   values into the transfer array before invoking event_schedule. */

    transfer[EVENT_TIME] = time_of_event;
    transfer[EVENT_TYPE] = type_of_event;
    list_file(INCREASING, LIST_EVENT);
}
```

图 2.54 Simlib 函数 event_schedule

```
int event_cancel(int event_type)
{

/* Remove the first event of type event_type from the event list, leaving its
   attributes in transfer.  If something is cancelled, event_cancel returns 1;
   if no match is found, event_cancel returns 0. */

    struct       master *row, *ahead, *behind;
    static float high, low, value;

    /* If the event list is empty, do nothing and return 0. */

    if(list_size[LIST_EVENT] == 0) return 0;

    /* Search the event list. */

    row   = head[LIST_EVENT];
    low   = event_type - EPSILON;
    high  = event_type + EPSILON;
    value = (*row).value[EVENT_TYPE] ;

    while (((value <= low) || (value >= high)) && (row != tail[LIST_EVENT])) {
        row   = (*row).sr;
        value = (*row).value[EVENT_TYPE];
    }

    /* Check to see if this is the end of the event list. */

    if (row == tail[LIST_EVENT]) {

        /* Double check to see that this is a match. */

        if ((value > low) && (value < high)) {
            list_remove(LAST, LIST_EVENT);
```

图 2.55 Simlib 函数 event_cancel

```
            return 1;
        }

        else /* no match */
            return 0;
    }

    /* Check to see if this is the head of the list.  If it is at the head, then
       it MUST be a match. */

    if (row == head[LIST_EVENT]) {
        list_remove(FIRST, LIST_EVENT);
        return 1;
    }

    /* Else remove this event somewhere in the middle of the event list. */

    /* Update pointers. */

    ahead         = (*row).sr;
    behind        = (*row).pr;
    (*behind).sr = ahead;
    (*ahead).pr  = behind;

    /* Decrement the size of the event list. */

    list_size[LIST_EVENT]--;

    /* Copy and free memory. */

    free((char *)transfer);        /* Free the old transfer. */
    transfer = (*row).value;       /* Transfer the data. */
    free((char *)row);             /* Free the space vacated by row. */

    /* Update the area under the number-in-event-list curve. */

    timest((float)list_size[LIST_EVENT], TIM_VAR + LIST_EVENT);
    return 1;
}
```

图 2.55 （续）

```
float sampst(float value, int variable)
{

/* Initialize, update, or report statistics on discrete-time processes:
   sum/average, max (default -1E30), min (default 1E30), number of observations
   for sampst variable "variable", where "variable":
       = 0 initializes accumulators
       > 0 updates sum, count, min, and max accumulators with new observation
       < 0 reports stats on variable "variable" and returns them in transfer:
           [1] = average of observations
           [2] = number of observations
           [3] = maximum of observations
           [4] = minimum of observations */

    static int   ivar, num_observations[SVAR_SIZE];
    static float max[SVAR_SIZE], min[SVAR_SIZE], sum[SVAR_SIZE];

    /* If the variable value is improper, stop the simulation. */

    if(!(variable >= -MAX_SVAR) && (variable <= MAX_SVAR)) {
        printf("\n%d is an improper value for a sampst variable at time %f\n",
            variable, sim_time);
        exit(1);
    }

    /* Execute the desired option. */
```

图 2.56　Simlib 函数 sampst

```
    if(variable > 0) { /* Update. */
        sum[variable] += value;
        if(value > max[variable]) max[variable] = value;
        if(value < min[variable]) min[variable] = value;
        num_observations[variable]++;
        return 0.0;
    }

    if(variable < 0) { /* Report summary statistics in transfer. */
        ivar        = -variable;
        transfer[2] = (float) num_observations[ivar];
        transfer[3] = max[ivar];
        transfer[4] = min[ivar];
        if(num_observations[ivar] == 0)
            transfer[1] = 0.0;
        else
            transfer[1] = sum[ivar] / transfer[2];
        return transfer[1];
    }

    /* Initialize the accumulators. */

    for(ivar=1; ivar <= MAX_SVAR; ++ivar) {
        sum[ivar]              = 0.0;
        max[ivar]              = -INFINITY;
        min[ivar]              =  INFINITY;
        num_observations[ivar] = 0;
    }
}
```

图 2.56 （续）

```
float timest(float value, int variable)
{

/* Initialize, update, or report statistics on continuous-time processes:
   integral/average, max (default -1E30), min (default 1E30)
   for timest variable "variable", where "variable":
       = 0 initializes counters
       > 0 updates area, min, and max accumulators with new level of variable
       < 0 reports stats on variable "variable" and returns them in transfer:
           [1] = time-average of variable updated to the time of this call
           [2] = maximum value variable has attained
           [3] = minimum value variable has attained
   Note that variables TIM_VAR + 1 through TVAR_SIZE are used for automatic
   record keeping on the length of lists 1 through MAX_LIST. */

    int          ivar;
    static float area[TVAR_SIZE], max[TVAR_SIZE], min[TVAR_SIZE],
                 preval[TVAR_SIZE], tlvc[TVAR_SIZE], treset;

    /* If the variable value is improper, stop the simulation. */

    if(!(variable >= -MAX_TVAR) && (variable <= MAX_TVAR)) {
        printf("\n%d is an improper value for a timest variable at time %f\n",
            variable, sim_time);
        exit(1);
    }

    /* Execute the desired option. */

    if(variable > 0) { /* Update. */
        area[variable] += (sim_time - tlvc[variable]) * preval[variable];
        if(value > max[variable]) max[variable] = value;
        if(value < min[variable]) min[variable] = value;
```

图 2.57 Simlib 函数 timest

```
        preval[variable] = value;
        tlvc[variable]   = sim_time;
        return 0.0;
    }

    if(variable < 0) { /* Report summary statistics in transfer. */
        ivar         = -variable;
        area[ivar]   += (sim_time - tlvc[ivar]) * preval[ivar];
        tlvc[ivar]   = sim_time;
        transfer[1]  = area[ivar] / (sim_time - treset);
        transfer[2]  = max[ivar];
        transfer[3]  = min[ivar];
        return transfer[1];
    }

    /* Initialize the accumulators. */

    for(ivar = 1; ivar <= MAX_TVAR; ++ivar) {
        area[ivar]   = 0.0;
        max[ivar]    = -INFINITY;
        min[ivar]    =  INFINITY;
        preval[ivar] = 0.0;
        tlvc[ivar]   = sim_time;
    }
    treset = sim_time;
}
```

图 2.57 （续）

```
float filest(int list)
{

/* Report statistics on the length of list "list" in transfer:
       [1] = time-average of list length updated to the time of this call
       [2] = maximum length list has attained
       [3] = minimum length list has attained
   This uses timest variable TIM_VAR + list. */

    return timest(0.0, -(TIM_VAR + list));
}
```

图 2.58 Simlib 函数 filest

```
void out_sampst(FILE *unit, int lowvar, int highvar)
{

/* Write sampst statistics for variables lowvar through highvar on file
   "unit". */

    int ivar, iatrr;

    if(lowvar>highvar || lowvar > MAX_SVAR || highvar > MAX_SVAR) return;

    fprintf(unit, "\n sampst                          Number");
    fprintf(unit, "\nvariable                          of");
    fprintf(unit, "\n number       Average            values          Maximum");
    fprintf(unit, "            Minimum");
    fprintf(unit, "\n___________________________________");
    fprintf(unit, "____________________________________");
    for(ivar = lowvar; ivar <= highvar; ++ivar) {
        fprintf(unit, "\n\n%5d", ivar);
        sampst(0.00, -ivar);
        for(iatrr = 1; iatrr <= 4; ++iatrr) pprint_out(unit, iatrr);
    }
    fprintf(unit, "\n___________________________________");
    fprintf(unit, "___________________________________\n\n\n");
}
```

图 2.59 Simlib 函数 out_sampst

```
void out_timest(FILE *unit, int lowvar, int highvar)
{

/* Write timest statistics for variables lowvar through highvar on file
   "unit". */

    int ivar, iatrr;

    if(lowvar > highvar || lowvar > TIM_VAR || highvar > TIM_VAR ) return;


    fprintf(unit, "\n  timest");
    fprintf(unit, "\n variable       Time");
    fprintf(unit, "\n  number       average          Maximum          Minimum");
    fprintf(unit, "\n___________________________________________________________");
    for(ivar = lowvar; ivar <= highvar; ++ivar) {
        fprintf(unit, "\n\n%5d", ivar);
        timest(0.00, -ivar);
        for(iatrr = 1; iatrr <= 3; ++iatrr) pprint_out(unit, iatrr);
    }
    fprintf(unit, "\n___________________________________________________________");
    fprintf(unit, "\n\n\n");
}
```

图 2.60 Simlib 函数 out_timest

```
void out_filest(FILE *unit, int lowlist, int highlist)
{

/* Write timest list-length statistics for lists lowlist through highlist on
   file "unit". */

    int list, iatrr;

    if(lowlist > highlist || lowlist > MAX_LIST || highlist > MAX_LIST) return;

    fprintf(unit, "\n  File         Time");
    fprintf(unit, "\n number       average          Maximum          Minimum");
    fprintf(unit, "\n___________________________________________________________");
    for(list = lowlist; list <= highlist; ++list) {
        fprintf(unit, "\n\n%5d", list);
        filest(list);
        for(iatrr = 1; iatrr <= 3; ++iatrr) pprint_out(unit, iatrr);
    }
    fprintf(unit, "\n___________________________________________________________");
    fprintf(unit, "\n\n\n");
}
```

图 2.61 Simlib 函数 out_filest

```
void pprint_out(FILE *unit, int i) /* Write ith entry in transfer to file
                                      "unit". */
{
    if(transfer[i] == -1e30 || transfer[i] == 1e30)
        fprintf(unit," %#15.6G ", 0.00);
    else
        fprintf(unit," %#15.6G ", transfer[i]);
}
```

图 2.62 Simlib 函数 pprint_out

```
float expon(float mean, int stream) /* Exponential variate generation
                                       function. */
{
    return -mean * log(lcgrand(stream));

}
```

图 2.63 Simlib 函数 expon

```
int random_integer(float prob_distrib[], int stream) /* Discrete-variate
                                                       generation function. */
{
    int   i;
    float u;

    u = lcgrand(stream);

    for (i = 1; u >= prob_distrib[i]; ++i)
        ;
    return i;
}
```

图 2.64 Simlib 函数 random_integer

```
float uniform(float a, float b, int stream) /* Uniform variate generation
                                              function. */
{
    return a + lcgrand(stream) * (b - a);
}
```

图 2.65 Simlib 函数 uniform

```
float erlang(int m, float mean, int stream)  /* Erlang variate generation
                                               function. */
{
    int   i;
    float mean_exponential, sum;

    mean_exponential = mean / m;
    sum = 0.0;
    for (i = 1; i <= m; ++i)
        sum += expon(mean_exponential, stream);
    return sum;
}
```

图 2.66 Simlib 函数 erlang

习题

以下习题尽可能使用 Simlib 做

2.1 对于第 2.4 节中用 Simlib 的单服务台队列，用你自己的变量表示服务台状态(忙或闲)，以代替服务台的哑元表，用 timest 代替 filest 来获得服务台利用率。如果你的计算机条件允许，比较原版本与修改版本仿真的时间。

2.2 对于第 2.5 节中的分时计算机模型，将 end-simulation 事件与 end-run 事件相结合。重新画出事件图，并改变和运行该简化的事件结构的程序。

2.3 对于第 2.5 节中的分时计算机模型，假设我们欲收集每个终端各自的平均响应时间及总响应时间。为此，请修改仿真，并只运行 $n=10$ 个终端的情形(提示：你必须添加另一个属性来表示作业的原始终端，你还需定义新的 sampst 变量)

2.4 对于第 2.6 节中的多出纳台银行模型，假设我们想知道队列中等待过的顾客最大值。按顺序完成以下各部分(每一部分是基于前一个步骤建立的)：

(a) 解释为什么把每个队列的最大值加起来不能得到它。

(b) 修改程序以收集该统计量，并输出结果。分别运行 $n=4$，5，6 和 7 个出纳台情形。

(c) 增加一项输出度量——服务台的利用率。由于有多个服务台，其利用率这里定义为忙碌服务台的时间平均数，再除以服务台个数。注意该值在 0 至 1 之间。

(d) 下面假设银行大厅大小只够容纳最多 25 个顾客在队列中(总数)。如果一个顾客到达后发现队列中已经共有 25 个顾客，则他/她将直接离去，他们的业务就丢失了；该情形称为阻塞，显然

是不幸的。修改程序，以反映出阻塞，其中容量 25 应作为输入参数读入。除了其他所有输出度量，还观测仿真过程中受阻的顾客总数。

2.5　在第 2.7 节中的制造系统模型中，修正报告生成器的关于按工件类型收集队列的总工件延误的微小误差，欲实现这点，可对每个工件添加一个属性以表示到目前为止队列中的累计延误。当工件离开系统时，计算 sampst 中的该值。用每个工作站中机器个数的“当前配置”，对此方法重新运行仿真。

2.6　在第 2.7 节中的制造系统模型中，估计工件在系统中的期望总平均时间，即三类工件在系统中的期望时间(队列中的延误时间加上处理时间)的加权平均，其中权值为每类工件发生的概率。(提示：你不需要计算机就能完成)

2.7　对于第 2.7 节中制造系统的原始配置，运行模型 100 个八小时工作日，但只使用后 90 天的数据来估计感兴趣的量。效果上，系统在第 10 天时的状态代表了仿真的初始条件。在开始收集数据之前的“预热”模型的概念在仿真实践中是常见的，见第 9.5.1 小节中讨论。(你也可观察图 2.57 中 Simlib 例程 timest 的代码，尤其注意变量 treset，以便理解连续时间统计是如何计算的)

2.8　对于第 2.7 节中的制造系统模型，给出一个能简化模型代码的不同的属性定义的建议。

2.9　用 Simlib 完成习题 1.15，流 1 用于到达间隔时间，流 2 用于服务台 1 的服务时间，流 3 用于服务台 2 的服务时间，流 4 用于移动时间。

2.10　用 Simlib 完成习题 1.22，流 1 用于机器正常时间，流 2 用于修理时间。

2.11　用 Simlib 完成习题 1.24，流 1 用于到达间隔时间，流 2 用于服务时间。注意该模型用表处理工具来仿真将是多么容易。

2.12　用 Simlib 完成习题 1.26，流 1 用于到达间隔时间，流 2 用于顾客类型的确定，流 3 用于类型 1 顾客的服务时间，流 4 用于类型 2 顾客的服务时间。

2.13　用 Simlib 完成习题 1.27，流 1 和流 2 分别用于常规顾客的到达间隔时间和服务时间，流 3 和流 4 分别用于快速顾客的到达间隔时间和服务时间。

2.14　用 Simlib 完成习题 1.28，流 1 用于常规车辆的到达间隔时间，流 2 用于所有车辆的服务时间。

2.15　用 Simlib 完成习题 1.30，流 1 用于到达间隔时间，流 2 用于检查时间，流 3 用于决定一辆汽车是否需要修理，流 4 用于修理时间。

2.16　对于第 1.5 节中的库存模型，假设交货延误是在 1 个月到 3 个月之间的均匀分布，从在某一时刻可能有 0 至 3 个未交付的订单。因此该公司在每月初基于(净)库存水平(在第 1.5 节中用 $I(t)$ 表示)加上已订购的库存来做它的订购决定。两者之和可能为正、零，或负。对 9 种库存策略的每一种，运行模型 120 个月，并估计每月期望的平均总成本以及存在未按期交货的期望时间比例。注意，持货成本与缺货成本仍然基于净库存水平。流 1 用于需求间隔时间，流 2 用于需求量，而流 3 用于交货延误。

2.17　习题 1.18 描述了第 1.5 节中库存系统的一个改动，其中货物有变质的可能。用 Simlib 完成该习题，增加考虑库存货物 LIFO(以及 FIFO)处理模式的情形。流分配同习题 2.16，并增加流 4 用于保质期。

2.18　对于第 2.5 节中的分时计算机模型，假设 CPU 不用轮转方式处理队列中的作业，而是从队列中选择之前通过 CPU 次数最少的作业。在有结的情况下，则规则是 FIFO(这等效于用到达队列的时间来解结)。运行本模型，其中终端数 $n=60$，作业完成个数为 1 000。

2.19　货船到达港口的间隔时间是均值为 1.25 天的 IID 指数随机变量。港口有一个码头，码头有两个停泊处，以及两个起重机用于卸货；当两个停泊处都被占用时，到达货船按照 FIFO 模式加入队列。起重机卸载一船货物的时间服从 0.5～1.5 天的均匀分布。如果港口只有一艘货船，则两个起重机同时卸载货物，且(剩余)卸载时间减半。当港口中有两艘货船时，则每个起重机负责一艘货船。如果两个起重机同时卸载一艘货船时又来了第二艘货船，则其中一个起重器立刻开始服务第二艘货船，第一货船的剩余服务时间翻倍。假设 0 时刻时港口中没有货船。运行一个 90 天的仿真，计算货船在港口中(包含在停泊处)的最长时间、最短时间及平均时间。此外，估计每个停泊处以及起重机的期望利用率。流 1 用于到达间隔时间，流 2 用于卸载时间。(本习题是 Russell(1976，第 134 页)例题的改编)

2.20　作业到达一个单 CPU 计算机设备的到达间隔时间是均值为 1 分钟的 IID 指数随机变量。每项作业到达时给出其所需处理时间的最大值，而后续作业的最大时间为 IID 指数型随机变量，均值为 1.1 分钟。然而，如果 m 是某项作业给出的最大处理时间，则其实际处理时间服从 $0.55m$～$1.05m$ 分钟的均匀分布。CPU 绝不会处理超过其给定最大时间的作业；若作业所需的处理时间超过其给定

的最大值，则其不进行服务而离开设备。仿真该计算机设备，直到有1 000项作业离开 CPU。考虑两种情形：(a) 队列中的作业按照 FIFO 方式处理；(b) 队列中的作业按照其给定的最大处理时间的递增顺序排列。对于每种情形，计算作业队列中平均延误时间和最大延误时间，在队列中的延误超过 5 分钟的作业比例，以及曾在队列中的作业最大值。流 1 用于到达间隔时间，流 2 用于最大处理时间，而流 3 用于实际处理时间。你推荐哪种运行策略？

2.21 在一个采矿场，卡车从三个挖掘机向一个碎石机运送矿石。卡车分配有特定的挖掘机，这样卡车总是在碎石机卸载之后回到自己所属的挖掘机。所用卡车有两种容量，20 吨和 50 吨。卡车的容量直接影响其在挖掘机处的装载时间、到碎石机的运送时间、在碎石机处的卸载时间，以及从碎石机返回自己挖掘机的回程时间，如表 2.26 所示(所有时间的单位：分钟)。

表 2.26

	20 吨型卡车	50 吨型卡车
装载	指数分布，均值为 5	指数分布，均值为 10
运送	常量 2.5	常量 3
卸载	指数分布，均值为 2	指数分布，均值为 4
回程	常量 1.5	常量 2

每个挖掘机配有两台 20 吨型卡车和一台 50 吨型卡车。挖掘机队列均为 FIFO，碎石机队列则按照卡车容量递减顺序排列，在有结(容量相同)的情况下是 FIFO 规则。假设时刻 0 时所有卡车分别在各自挖掘机处，50 吨型卡车正准备装载。运行仿真模型 8 小时，估计每个挖掘机以及碎石机队列中期望时间平均数，还估计所有四台设备的期望利用率。流 1 和流 2 分别用于 20 吨和 50 吨型卡车的装载时间，流 3 和流 4 分别用于两种类型卡车的卸载时间。(本习题取自 Pritsker(1995，第 153-158 页))

2.22 单一 CPU 的批量作业的计算机设备，早晨 7 点开门，午夜关门，但继续运行直到午夜尚存的所有作业处理完为止。假设作业到达设备的间隔时间服从均值为 1.91 分钟的指数分布。作业可要求快速(类别 4)、正常(类别 3)、延缓(类别 2)和简便(类别 1)的服务；类别发生的概率分别为：0.05、0.50、0.30、0.15。当 CPU 空闲时，将处理尚存的最高类别(优先数)的作业，同类作业的规则是 FIFO。CPU 处理 4、3、2、1 各类作业所需的时间为 3-Erlang 随机变量(参见第 2.7 节)，相应的均值分别为 0.25、1.00、1.50、3.00 分钟。按以下情形分别仿真该计算机设备。

(a) 一个正在处理的作业不被新到达的更高类别的作业取代。

(b) 如果类别 i 的作业正在处理，且类别 $j(j>i)$的作业到达，则到达作业将取代正在处理的作业。被取代的作业进入队列等待，并在同类别作业中赋予最高优先级，只是在将来的某个时间，其剩余服务时间必须完成。对于每个类别，估计队列中作业的期望时间平均数，以及队列中期望平均延误时间。还估计 CPU 忙碌时间的期望比例，以及 CPU 花费在每类作业上的忙碌时间的期望比例。注意，为方便起见，每类队列有一个表，还有一个输入参数，对情形(a)参数设为 0、对情形(b)参数设为 1。流 1 用于到达间隔时间，流 2 用于作业类别的确定，流 3、4、5、6 分别用于类别 4、3、2、1 的处理时间。

2.23 非洲的一个港口为油轮装载原油水运，港口设备可同时给三个油轮装载。每隔 11±7 小时有三种类型的油轮抵达港口(本题中所有时间都带有“±”区间，表示在该区间内均匀分布)。油轮各种类型的相对频率及其所需装载时间如表 2.27 所示。

表 2.27

类型	相对频率	装载时间(单位：小时)
1	0.25	18±2
2	0.25	24±4
3	0.50	36±4

港口有一个拖船。各类油轮都需要拖船将其从港口拖往停泊处，之后再从停泊处拖入港口。当拖船可用时，任意停泊和驶离活动都需要大约 1 小时。而当拖船不拖油船时，来往于港口和停泊处之间需要 0.25 小时。当拖船完成一次停靠活动时，若驶离队列非空，则拖船拖走该队列中第一艘油轮。若驶离队列为空而港口队列非空，则拖船将驶回港口并开始拖送港口队列中的第一艘油轮停

泊(若两个队列均空，则拖船将在停泊处闲置)。当拖船完成一次驶离活动时，若港口队列非空且停泊处可用，则将拖送港口队列中的第一艘油轮去停泊。否则，拖船驶回停泊处；若驶离队列非空，则继续拖送该队列中的第一艘油轮至港口。如果驶离队列为空，则拖船在停泊处闲置。

然而仿真由于以下情况将变得进一步复杂，即该地区经常有持续 4±2 小时的风暴。一次风暴的结束和下一次风暴开始之间的时间是一个指数型随机变量，均值为 48 小时。拖船在风暴进行时不会开始一个新的活动，但会完成已在进行中的活动(停泊处在风暴中仍将工作)。如果拖船在风暴开始时正在不带油轮从停泊处驶向港口，则它将即刻掉头驶回停泊处。

运行仿真模型一个年周期(8760 小时)，并估计：

(a) 拖船闲置、不带油轮行驶，以及忙于停泊或驶离活动的期望时间比例。

(b) 每个停泊处未占用、被占用但不装载、装载的期望时间比例。

(c) 驶离队列以及港口队列中油轮的期望的时间平均数。

(d) 每类油轮期望的平均在港停留时间。

流 1 用于到达间隔时间，流 2 用于油轮类型，流 3 用于装载时间，流 4 用于风暴持续时间，流 5 用于一次风暴结束与下一次风暴开始之间的间隔时间。

有个托运商在考虑承包一份从该港口向英国运油的合同，托运商决定该任务必须交给五艘某一特定类型的油轮以满足该合同要求。这些油轮在港口装载原油需要 21±3 小时。装载并驶离港口后，这些油轮驶向英国，卸载原油，再返回港口重装原油，以此类推。往返路程时间，包含卸载时间，估计为 240±24 小时。重新运行仿真，另外并估计建议的附加油轮的期望的平均港内停留时间。假设在时刻 0 这五艘附加油轮在港口队列中。与以前一样指派随机数流，增加流 6，用于港口的原油装载时间，流 7 用于这些新油轮的来回行驶时间。(该习题改自 Schriber 的问题之一，1974，第 329 页)

2.24　在习题 2.23 中，假设拖船有一个收发报机给出港口内每艘油轮的位置及状态。因此，拖船改变其运行策略如下。如果拖船空载正在从港口驶向停泊处，且行驶距离不到一半时新到达一艘新的油轮，则其调头回去拖送该油轮。如果拖船空载从停泊处驶向港口，且行驶距离不到一半时，一艘油轮刚好完成其装载任务，则其调头回去拖送该装载完的油轮。在新的运行策略下运行该仿真，参数相同，流指派同前。

2.25　在习题 2.24 基础上再假设，如果拖船正空载从港口驶向停靠处，新油轮到达时驶离队列为空，此时不管拖船位置在哪，它将调头回去拖送该油轮。在新的运行策略下运行该仿真，参数相同，流指派同前。

2.26　干洗机按照以下方式清洗两件套的套装。套装到达间隔时间服从均值为 10 分钟的指数分布，且最初均由服务台 1 处理，可能在先进先出队列中等待一段时间，参见图 2.67。服务台 1 完成后，套装的一件(夹克)进入服务台 2，另一件(裤子)进入服务台 3。在服务台 2 中夹克损坏的概率为 0.05，而在服务台 3 中裤子损坏的概率为 0.10。离开服务台 2 的夹克进入服务台 4 的一个队列中；离开服务台 3 的裤子进入服务台 4 的另一个队列中。若服务台 4 空闲且来自同一套装的两件均已在队列中，则该服务台将其匹配并重组套装，将其复原。如果重组套装的两件均未损坏，则套装返回给顾客。如果两件中一件(或两件)有损坏，套装进入客户关系部(服务台 5)。假设所有的服务时间均服从指数分布，其均值(以分钟为单位)和使用指定的流分配如表 2.28 所示。

表　2.28

服务台编号	平均服务时间(单位：分钟)	流
1	6	1
2	4	2
3	5	3
4	5(未损坏)	4
4	8(已损坏)	5
5	12	6

此外，流 7 用于到达间隔时间，流 8 和流 9 分别用于服务台 2 和 3 中的套装件是否损坏。系统最初状态为空且闲，运行整 12 小时。分别观察每类输出(损坏的与未损坏的)在系统中平均值与最大值，每个队列长度的平均值和最大值，以及每个服务台的利用率。若到达速率翻倍(即到达间隔时

间的均值是 5 分钟而不是由 10 分钟)将会发生什么样的情况？在此情形下，如果你能向系统中添加另外一人来帮助 5 项任务中的某一项，应是哪儿？

2.27　一个排队系统有两个串行服务台(A 和 B)，以及两类顾客(1 和 2)。顾客到达系统后很快便可确定其类别。类型 1 顾客的到达概率为 0.6。然而，到达顾客可能阻塞，即如果服务台 A 的队列过长，则该顾客可能并不真正进入系统。特别是，假设一个到达顾客发现 A 的队列中已经有 $m(m\geqslant 0)$ 个其他顾客，则其进入系统的概率为 $1/(m+1)$，不管他属于哪类顾客(1 或 2)。因此，举个例子，若到达顾客发现 A 的队列中无人(即 $m=0$)，则他一定进入系统(概率为 $1/(0+1)=1$)；而到达顾客若发现 A 的队列中已经有 5 个其他顾客了，则其进入系统的概率为 1/6。所有顾客都由 A 服务(若顾客到达时 A 忙，则顾客进入 FIFO 队列)。在 A 的服务完成后，类型 1 顾客离开系统，而类型 2 顾客由 B 服务(若 B 忙，则类型 2 顾客在 FIFO 队列中等待)。计算每类顾客花在系统中的平均总时间，以及阻塞的顾客个数。另外计算每个队列的时间平均长度与最大长度，以及两个服务台的利用率。假设所有到达间隔时间及服务时间均服从指数分布，参数如下：

- 平均到达间隔时间(任意类型顾客)为1 分钟。
- 服务台 A 的平均服务时间(不管顾客类型)为 0.8 分钟。
- 服务台 B 的平均服务时间为 1.2 分钟。

系统初始状态为空且闲，运行到离开系统的顾客(两类)数达到 1 000。流 1 用于顾客类型，流 2 用于顾客是否阻塞，流 3 用于到达间隔时间，流 4 用于 A 的服务时间(两类顾客)，流 5 用于 B 的服务时间。

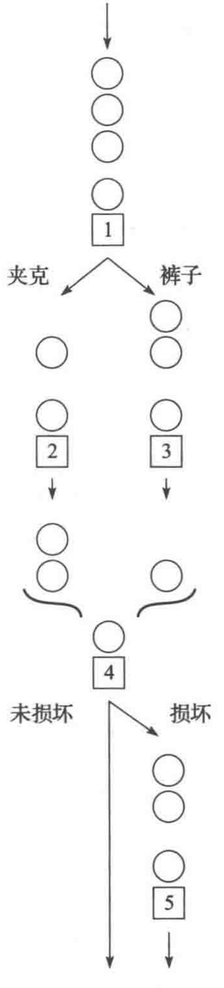


图 2.67　干洗店运行

2.28　一个过时的计算机按批多处理模式运行，意思是，在某一时刻启动多项作业(最大固定为 $k=4$)，同时运行它们，但直到这批的所有作业都完成后才能启动新的作业。在一批内，每项作业有自己的完成时间，完成后离开 CPU。作业有三个优先级，一级作业优先级最高，三级作业优先级最低。当 CPU 完成一批中的最后一项作业时，首先在队列中搜索一级队列的作业，并尽可能多取，最多取 k 项。若一级队列中的作业小于 k 项，则取出尽可能多的二级作业，使取出的一级和二级作业的总数不超过 k。若这批仍有余量，则 CPU 移到三级作业队列。若所有队列中等待的作业的总数小于 k，则 CPU 将其全部取出，并开始运行这非满的批；当前批全部完成之前不得开始任何后续到达的作业。如果队列中没有作业等待，则 CPU 变为闲，且下一到达作业会启动 CPU 运行作业项数为 1 的批。注意，当一批作业开始运行时，同一批可能有多种级别的作业一起运行。

在一个级别的队列中，作业取出的顺序可以是先进先出(FIFO)，也可以是最短作业先出(SJF)；所写的仿真要求仅通过修改一个输入参数来处理两种队列规则(也就是说，作业的服务要求应该在其到达时生成，并与其到达时间一起存储于队列中。对于 FIFO，不一定非要这样做，但这样可以简化编程)。级别 i 的作业的服务需求服从常数 $a(i)$ 与 $b(i)$ 分钟之间的均匀分布。每个级别有其独立的到达过程，即两个连续的级别 i 的作业之间的到达间隔时间服从均值为 $r(i)$ 分钟的指数分布。因此，在仿真的任意给定时刻，应安排三个独立的到达，每个级别各一个。如果作业到达时发现 CPU 忙，则根据是选 FIFO 还是 SJF，进入该级队列适当位置。发现 CPU 闲的作业到达立刻开始服务，该批作业数为 1。各参数如表 2.29 所示。

表　2.29

i	$r(i)$	$a(i)$	$r(i)$
1	0.2	0.05	0.11
2	1.6	0.94	1.83
3	5.4	4.00	8.00

最初系统为空且闲，运行仿真正好720分钟。对每个队列，计算平均、最小及最大延误时间，以及时间平均长度与最大队列长度。此外，计算CPU的利用率，此处定义为CPU忙碌时间的比例，而不管作业运行的个数。最后，计算CPU中作业运行的时间平均数(这里，当CPU空闲时，运行作业数视作0)。流1、2和3分别用于级别1、2和3的作业到达间隔时间，流4、5和6分别用于它们各自的服务时间。假设欲升级硬件使k增大为6，请问这样值吗?

2.29 考虑排队系统，单一队列进入，固定数目$n=5$个并行服务台。顾客到达间隔服从均值为5(所有时间的单位均为分钟)指数分布。顾客到达发现有服务台空闲，则直接进入服务；若有多个服务台空闲，则选择最左边的；而当到达发现所有服务台均忙碌时则进入队尾。当顾客(最初)进入服务时，其服务需求服从$a=2$与$b=2.8$之间的均匀分布，然而当其初始服务完成时，她可能对服务“不满意”，发生概率为$p=0.2$。若服务满意，则顾客直接离开系统；若不满意，则该顾客还需要进一步服务。服务是否满意是在服务完成时确定。如果不满意的服务完成且队列中没有其他顾客在等待，该不满意顾客立刻在同一个服务台开始另一次服务时间。反之，如果不满意的服务完成时队列有人，则该不满意顾客必须进入队列等待(规则二选一，如下所述)，该服务台继续服务队列中下一等待服务的第一个顾客。每当一个顾客重新进入服务时，其服务时间与不满意概率均小一些；特别地，已有i次(不满意)服务的顾客，其下一次服务时间服从$a/(i+1)$与$b/(i+1)$之间的均匀分布，该下一次服务的不满意概率为$p/(i+1)$。理论上讲，要使服务的顾客最终满意的服务次数没有上限。

对不满意的顾客来说，当队列中有他人在等待时，有两种可能的规则供选择，程序的编写使得重新规定一个输入参数就可从规则(i)变到(ii)：

(i) 刚完成一次不满意的服务的顾客进入队尾；

(ii) 刚完成一次不满意的服务的顾客重新入队，使得来自队列(的前面)的下一个人是已经得到服务次数最多的顾客；在有结的情况下，规则是FIFO。该规则是为了公平性及有效性，因为不满意次数多的顾客更希望尽快服务，从而下一次服务满意的可能性更大。

最初系统为空且闲，仿真运行正好480分钟。计算系统中的平均、最大总时间(包括队列中的所有延误及顾客的服务时间)，以及仿真期离开系统的满意的顾客数。同时计算队列的平均与最大长度，服务台忙的时间平均数与最大数。流1用于到达间隔时间，流2用于所有服务时间，流3用于每项服务是否满意。

2.30 大州大学的学生中心餐厅正试图改善11：30～13：00的午餐高峰时段的服务。顾客成组到达，每组人数为1、2、3、4，概率分别为0.5、0.3、0.1、0.1。组间的到达间隔时间服从均值为30秒的指数分布。一开始，系统为空且闲，运行时长为90分钟周期。每位到达顾客，不管是一个人还是和别人成一组，都按照以下三种路线之一进入餐厅(各组通常在到达之后分散开来)：

- 热食服务，然后饮料，最后收银台；
- 特色三明治吧，然后饮料，最后收银台；
- (只选)饮料，然后收银台。

各条路线的概率分别为0.80、0.15、0.05；参见图2.68。在热食柜台以及特色三明治吧，每次服务一个顾客(尽管实际可能有一个或两个工作人员，下面将讨论)。饮料台是自助的，并假设此处始终不需要排队；这等价于认为饮料台有无限多个服务口。收银台共有两个或三个(见下面)，各有自己的队列，且不换队；到达收银台的顾客只是简单的选择最短队列。模型中的所有队列均为FIFO。

在图2.68中，ST表示某站的服务时间，ACT表示访问某站后的累计(将来)收银时间；记号$U(a, b)$表示相应量服从a和b秒之间的均匀分布。例如，路线1的一个顾客首先来到热食站，必要时进入那儿的队列，在那儿接受服务，时间服从50～120秒之间的均匀分布，记下(将来)收银时间的当前部分值，该值均匀分布于20～40秒之间；接下来该顾客选择饮料，花费时间在5～20秒之间均匀分布，并在(将来)收银时间上累加一个5～10秒均匀分布的附加时间。因此，该顾客在收银台的服务需求时间为$U(20, 40)$和$U(5, 10)$随机变量之和，表示他在热食及饮料站“取食品”的结账时间。

报告系统性能的以下度量：

- 热食、特色三明治以及收银台(不考虑哪个收银台)队列中的平均延误时间和最大延误时间；
- 热食和特色三明治(各自)队列中时间平均与最大数，以及所有收银台队列中的时间平均总数与最大总数；

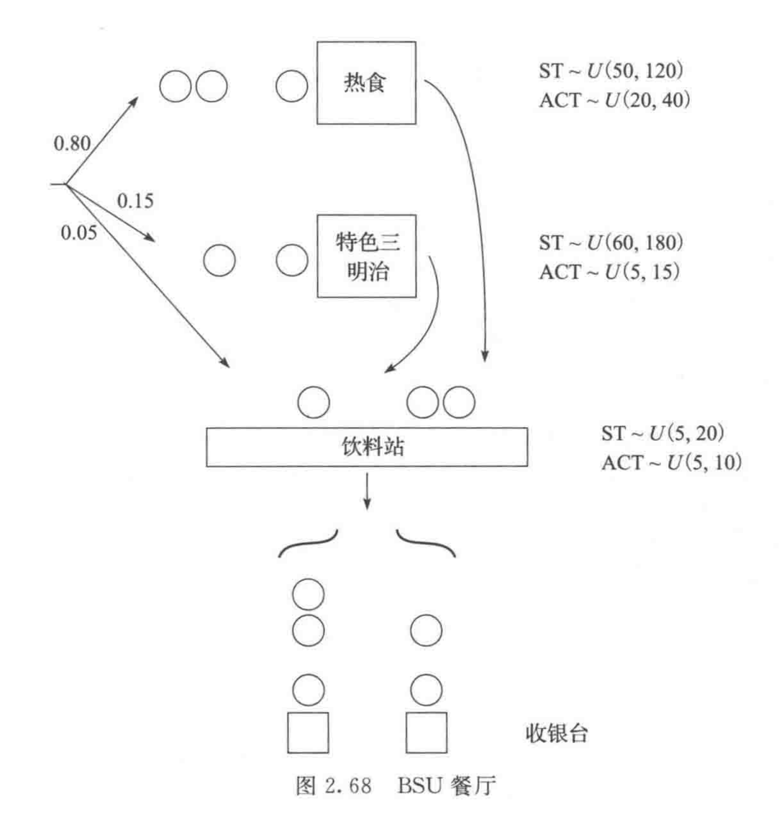


图 2.68　BSU 餐厅

- 三类顾客中每类(各自)在所有队列中的平均总延误时间和最大总延误时间；
- 所有顾客整体的平均总延误时间，即每个顾客平均总延误时间乘以各自的出现概率的加权平均；
- 整个系统中顾客的时间平均总数与最大总数。

关于系统运作还有几个问题。出于安全原因，必须至少有 2 个收银台，且其最大值为 3。同时，在热食与特色三明治站分别必须至少有一个工作人员服务。因此，雇员数最少为 4；运行它作为“基础方案”模型。

接下来，考虑增加雇员，有下面几种方式。

(a) 5 个雇员，以以下方式使用增加的那个人：

(i) 作为第三个收银员。

(ii) 帮助热食站。在这种情况下，仍是每次服务一个顾客，但其服务时间减半，即为 25～60 秒之间的均匀分布。

(iii) 帮助特色三明治站，意味着，仍是每次服务一个顾客，服务时间为 30～90 秒之间的均匀分布。

(b) 6 个雇员，按以下配置之一：

(i) 2 个在收银台，热食站与特色三明治站每处 2 个。

(ii) 3 个在收银台，2 个在热食站，1 个在特色三明治站。

(iii) 3 个在收银台，1 个在热食站，2 个在特色三明治站。

(c) 7 个雇员，3 个在收银台，2 个在热食站，2 个在特色三明治站。

对所有 7 种扩展可能性进行仿真，并提出在不同水平的雇员数下最好的雇员配置建议。在所有情形中，流 1 用于小组间的到达间隔时间，流 2 用于小组人数，流 3 用于个体路线选择，流 4、5、6 分别用于热食、三明治和饮料站的 ST，流 7、8、9 分别用于三个站各自的 ACT。

2.31　Consolidated Corkscrews(CC)是一个多国制造商，生产用于重型、高速的精密碳钢开塞钻。每个开塞钻在金属车床上加工。为了满足顾客对其产品的不断增长的需求，CC 正在计划一个有 6 个车床的新车间。然而他们并不确定是否应该建这个新车间，以及维护部门应该如何装备。每台车床有独自的操作员，当车床故障停机时，操作员还负责修理车床。车床操作的可靠性数据表明，车床“正常工作”的时间服从均值 75 分钟的指数分布。当车床故障停机时，其操作员立即呼叫工具

间索要修理工具箱。车间共有固定数目工具箱 m 个，因此当操作员呼叫工具间时，可能有也可能没有工具箱。如果没有工具箱可用，则操作员需求列入一个 FIFO 队列且必须等待轮到他或她的工具箱；当后来有工具箱可用时，它被放到传送带上，并于 t_i 分钟后到达车床 i，其中 t_i 可能取决于的请求工具箱的车床编号 i。若有工具箱可用，则立刻将它放到传送带上并于 t_i 分钟后到达故障停机的车床 i；在这种情况下，操作员的队列延误为 0。当故障停机车床的操作员收到工具箱时，他或她立即开始修理，花费的时间服从均值为 15 分钟的 3-厄兰分布。当修理完成时，车床重新正常工作，工具箱送回工具间，如果它是从车床 i 处返回，则 t_i 分钟后到达。初始时，假设所有车床均正常工作，且均为“刚修理完”的状态，并假设所有 m 个工具箱都在工具间里。CC 想知道连续 24 小时的一天中计划的车间运作情况，具体考察：

- 六台车床每台故障停机的时间比例；
- 车床故障停机的时间平均数；
- 工具间中工具箱空闲的时间平均数；
- 请求工具箱的操作员在队列中的平均延误时间。

有如下两个主要问题要说明。

(a) 车间应该如何布置？目前考虑两种布置。

(1) 在线性设计中(参见图 2.69)，车床以直线方式放置，工具间在最左端，一条传送工具箱的传送带可达到所有车床。在这种情况下，$t_i=2i$(分钟)，$i=1, 2, \cdots, 6$。

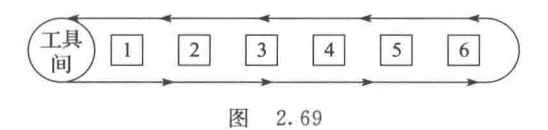


图　2.69

(2) 在环形设计中，车床围绕工具间放置(参见图 2.70)，每台车床有它自己的传送带到工具间；这里，对于所有车床 i，$t_i=3$。这是一个较贵的设计，但可减少工具箱的传送时间。

(b) 应该准备多少个工具箱？因为工具箱相当贵，CC 不希望采购多余的工具箱。

进行必要的仿真，向 CC 给出问题(a)和(b)的建议。在所有情形中，流 1 用于车床的工作时间，流 2 用于修理时间。

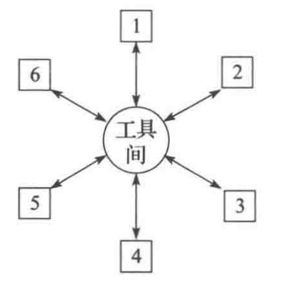


图 2.70　环形设计

2.32　喷气式飞机的引擎必须定期检查，且必要时给予维修。大型机场的检修设备可以处理七种不同类型的喷气式飞机，如表 2.30 所述。类型 i(其中 $i=1, 2, \cdots, 7$)的的飞机相继到达的时间间隔服从均值为 $a(i)$ 的指数分布，如表 2.30 所示；所有时间单位为天。共有 n 个并行的服务站，每个服务站依次处理一架飞机所有引擎的检查和修理工作，但一次只能处理一个引擎。例如，一架类型 2 的飞机有 3 个引擎，当它进入服务时，每个引擎必须经历一个完整的检查和修理过程(如下所述)之后，再进行下一个引擎的服务；所有三个引擎在飞机离开服务站之前都必须检查并且(必要时)修理。每个服务站能处理任意类型的飞机。通常，到达飞机发现有空闲的服务站，则其直接进入服务；而到达飞机发现所有服务站均被占用，则必须进入单一队列。

表　2.30

飞机类型(i)	引擎个数	$a(i)$	$A(i)$	$B(i)$	$p(i)$	$r(i)$	$c(i)$
1	4	8.1	0.7	2.1	0.30	2.1	2.1
2	3	2.9	0.9	1.8	0.26	1.8	1.7
3	2	3.6	0.8	1.6	0.18	1.6	1.0
4*	4	8.4	1.9	2.8	0.12	3.1	3.9
5	4	10.9	0.7	2.2	0.36	2.2	1.4
6	2	6.7	0.9	1.7	0.14	1.7	1.1
7*	3	3.0	1.6	2.0	0.21	2.8	3.7

七种类型的飞机中，有两种属于宽体机(上表中用“*”号标识)，而另外五种属于普通机体。排

队规则有如下两种：

(i) 所有类型飞机混在一起在同一队列中简单的 FIFO；

(ii) 宽体机具有非抢占式优先，宽体机和普通机内部的规则是 FIFO。

飞机上的每个引擎(独立)，均发生以下过程(i 表示飞机类型)：

- 引擎初始检查，所需时间服从 $A(i)\sim B(i)$之间的均匀分布。
- 决定是否需要修理，需要修理的概率为 $p(i)$。如果不需要修理，开始飞机下一个引擎的检查；或者如果这是最后一个引擎，则飞机离开检修站。
- 如果需要修理，则进行修理，所需时间服从均值为 $r(i)$的 2-厄兰随机变量。
- 修理后，再做一遍检查，所需时间服从 $A(i)/2\sim B(i)/2$ 之间的均匀分布(即初始检查时间的一半，因为已做拆除)。引擎还需进一步修理的概率为 $p(i)/2$。
- 如果初始修理成功，则该引擎检修结束。若引擎未通过检查，引擎还需进一步修理，所需时间服从均值为 $r(i)/2$ 的 2-厄兰分布。修理之后再检查一遍，检查时间服从 $A(i)/2\sim B(i)/2$ 之间的均匀分布；未通过检查的概率为 $p(i)/2$，若仍需修理，则还需修理时间服从均值为$r(i)/2$的 2-厄兰分布。该过程不断继续，直到引擎最终通过检查为止。平均修理时间一直为 $r(i)/2$，失败的概率一直为 $p(i)/2$，且检查时间一直在 $A(i)/2\sim B(i)/2$之间。

一架类型 i 的飞机停飞，即在队列中以及在服务中，每天(全天)需要的花费为 $c(i)$。一般的概念是，研究总(各类飞机求和)平均每日停机花费如何依赖于服务站的个数 n。最初系统为空且闲，运行仿真 365 天连续不断。观察每类飞机在队列中的平均延误时间、所有类型飞机在队列中总平均延误时间、队列中飞机时间平均数、每类停飞飞机各自的时间平均数，以及所有飞机加在一起的总平均每日停机花费。尝试不同的 n 值来观察系统行为特性。给出选择 n 的建议，同时给出上述队列规则(i)和(ii)中哪一种会划算。流 1～7 分别用于七种类型飞机各自的到达间隔时间，流 8～14用于各自的检查时间(初始或再修的)，流 15～21 用于相应的需要修理的概率，流 22～28 用于各自的修理时间(第一次或后续的)。

作为上述布局的一种替代方案，考虑将宽体机和普通机的整个服务分开。也就是说，在 n 个检修站中，假设有 n_2 个站，所有的宽体机送到那儿(宽体机用单一队列进入所有 n_2 个站)，剩余的 $n_1=n-n_2$个站用于普通机，参见图 2.71。你认为这种替代布局会更好吗？为什么？使用同前的参数和流分配。

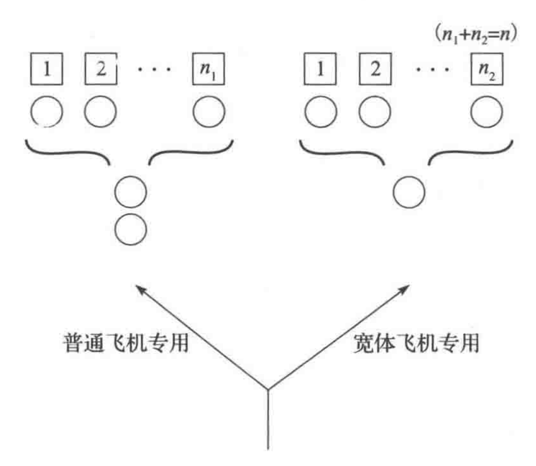


图 2.71　飞机修理设施的替代布局

2.33　写一个 C 语言函数 “delete”，从表 “list” 中删除属性 “attribute” 的值为 “value” (用浮点值表示一个整数，如 5.0 表示 5)的(逻辑上的)第一条记录。将被删记录的属性存入 transfer 数组中。要删除一条所要求的记录，应该执行形式为 “delete(list，value，attribute)” 的语句。如果出现错误情形(比如，表中有非匹配记录)，则返回值 0；否则返回值 1。另外，通过调用 timest 来更新表 “list” 的统计(参见附录 2A 中函数 remove 的代码)。

2.34　写一个 C 语言函数 “insert”，使用 2.8 节中讨论的中间指针算法，向事件表中插入一条新的事件记录。如果两条事件记录具有相同的事件时间，则具有最小编号事件类型的事件优先。

2.35　对于第 2.6 节中的银行模型，假设一个顾客在队列中等待一定时间后，该顾客可能选择未服务而

离去，这称为放弃。假设顾客在考虑放弃之前将在队列中等待的时间服从 5～10 之间的均匀分布；若顾客在队列中等待的实际时间超过该值，则顾客实际离开系统的概率如表 2.31 所示。

表 2.31

超时时在队列中的位置	1	2	3	≥4
放弃的概率	0.00	0.25	0.50	1.00

使用习题 2.33 的函数 “delete”，运行五个出纳台的仿真模型，并估计(除前面要估计的之外还需估计)放弃的顾客的期望比例，以及放弃顾客在队列中的期望平均延误时间。使用与 2.6 节中一样的流分配，此外流 3 用于在考虑放弃之前顾客将在队列中等待的时间，流 4 用于如果超时了要确定他或她是否真的放弃。

2.36 一台电梯为一个五层的办公楼服务。人们以均值为 1 分钟的指数分布到达间隔时间来到地面层(第 1 层)。无论在哪，人们上每一层楼的概率为 0.25。上一层楼电梯需要 15 秒钟。但是。假设电梯在某一层无载客卸客时间。人们在某一层的停留时间服从 15 到 120 分钟之间的均匀分布。当某人离开 i 层(其中 $i=2, 3, 4, 5$)，她或他以 0.7 的概率去一层，以 0.1 的概率去其他三层的任何一层。电梯能载客 6 人，并从一层启动。电梯到达的时候，如果在某一层电梯无法装载所有等待电梯的人，则超额的人留在队列中。下到一楼的人立即离开大楼。以下控制逻辑同样应用于该电梯：

- 当电梯上升时，如果当前乘客想去更高的楼层，或者更高楼层有人想乘电梯，则继续上升。
- 当电梯下降时，若它至少有一个乘客或者更低楼层有人想乘电梯，则继续下降。
- 如果电梯正在第 i 层(其中 $i=2, 3, 4$)，且将要上升(或下降)，则不会立即搭乘在那层想要下降(或上升)的乘客。
- 当电梯闲时，它的驻地是第一层。
- 电梯在每层要决定下一层去哪。在楼层之间电梯不会改变方向。

使用以下流分配。

流 1：人们到达大楼的间隔时间。

流 2：下一层去向决定(到达原定层时生成)。

流 3：在某一层停留的时长(到达某层时生成)。

运行仿真 20 小时，并搜集统计：

(a) 在每一层，每个方向(如果合适的话)队列中的平均延误时间；

(b) 所有层所有乘客队列中个人延误的平均值时间；

(c) 电梯搭载乘客运行、空载运行，以及闲置(在第一层)的时间比例；

(d) 电梯中平均人数与最多人数；

(e) 在每一层，由于人满而不能搭乘电梯的人员比例。

若电梯的驻地为第三层，重新运行仿真。请问驻地为哪一层时，平均延误时间(输出统计(b))最小?

2.37 拉煤列车以均值为 10 小时的独立指数到达间隔时间到达卸货设施。如果列车发现系统闲，则立即卸货。列车卸货时间是独立均匀分布在 3.5 到 4.5 小时之间。如果列车到达系统忙，它进入 FIFO 队列。

由于铁路会出现所谓“空档”的情况，所以问题变得复杂起来。特别是，一个铁路工人只能工作 12 小时，当工人不在时，列车就不能卸货。当列车到来时，铁路工人剩下的时间(12 小时内的)独立均匀分布在 6 到 11 小时之间。当工人工作了 12 小时，他会立即离去，并通知替换工人。替换工人接到通知与实际到达之间的时间数量是独立均匀分布在 2.5 到 3.5 小时之间。

如果火车正在卸货时工人空档了，那么卸货暂停，直到替换工人到达。当火车在排队时，工人空档了，那么替代工人到来之前，火车不能离队。于是，即使队列中有一个或多个列车，卸载装备仍处于闲。

运行此仿真 720 小时(30 天)并收集如下统计数据：

(a) 列车在系统中的平均和最大时间；

(b) 卸载设备处于忙碌，空闲，和“空档”时间的比例；

(c) 队列中列车的平均和最大数；

(d) 列车“空档”0 次，1 次，2 次的比例。

注意，如果列车在队列中工人空档，必须访问该列车的记录(这个列车可能在队列中的任意位置)。使用习题 2.33 中的 C 语言函数“delete”。

2.38 考虑一个图 2.72 所示的汽车租赁系统，所有距离单位为英里。租车的人分别以每小时 14，10，和 24 人的速率和独立指数到达间隔时间来到地点 $i(i=1,\ 2,\ 3)$。每一个地点都有一个可以无限容量的 FIFO 队列。有一辆公交车，容量为 20 人，速度为每小时 30 英里。公交车最初停在地点 3(汽车租赁)，并立即以逆时针方向离开。所有到站的人们都要去租赁汽车，即地点 3。而所有到达汽车租赁的人希望到地点 1 和 2，概率分别为 0.583 和 0.417。当汽车到达一个站，应遵循以下的规则：

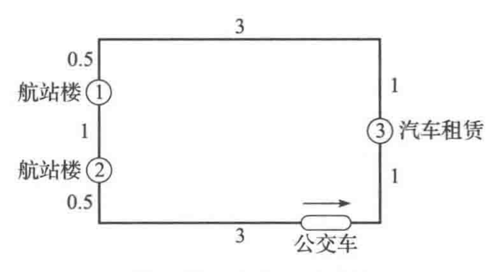


图 2.72　汽车租赁系统

- 人们应以先入先出的规则下车，下车一人的时间均匀分布在 16～24 秒之间。
- 然后载客至满，每个人上车时间均匀分布在 15～25 秒之间。
- 公交车在每个地点至少花 5 分钟。如果 5 分钟后没有乘客上下，则公交车立即离去。

运行此仿真 80 小时，收集如下统计数据：

(a) 每个队列中的平均和最大数；

(b) 每个队列中的平均和最大延误时间；

(c) 公交车上的平均和最大人数；

(d) 公交车停在在每一个地点的平均，最大和最小时间；

(e) 汽车转一圈(从汽车租赁地点开出到下一个这样开出)的平均，最大和最小时间；

(f) 一个人到到达地在系统中所用的平均，最大和最小时间。

使用下列随机数流分配：

i，在地点 i 的到达时间间隔($i=1,\ 2,\ 3$)；

4，下人时间；

5，上人时间；

6，确定到达汽车租赁点的顾客的目的地。

第3章

仿 真 软 件

3.1 引言

在学习第1章和第2章中的仿真例子时，读者可能注意到了大多数离散事件仿真模型编程中所需要的几个要素，它包括如下几个。

- 产生随机数，即均匀概率分布$U(0, 1)$的观测值；
- 产生一个特定概率分布(例如，指数分布)的随机变量；
- 推进仿真时间；
- 从事件表中确定下一事件，并将控制权转交给适当的代码块；
- 向一个表添加记录或从表中删除记录；
- 搜集输出统计信息并生成结果报告；
- 探测错误发生的条件。

事实上，这些特征对大多数仿真程序来说是通用的，这就导致了专用仿真软件包的开发。我们认为，近年来仿真之所以日益流行，主要原因是这些软件包的改进与其更易于使用。

在第3.2节我们将讨论使用仿真程序包建立仿真模型相对于使用编程语言(如C、C++或Java语言)的优势；在第3.3节我们给出仿真软件的分类，包括对通用的和面向应用的仿真软件包的讨论；第3.4节介绍仿真软件包的期望特性，包括动画；第3.5节给出两个流行的通用仿真软件包——Arena和Extend的简要描述，并分别利用这两个仿真软件包构建了一个小工厂的仿真模型；在第3.6节，我们介绍面向对象的仿真软件；最后，在第3.7节，我们介绍若干面向不同应用的仿真软件包。

刊物“OR/MS Today”往往定期地进行仿真软件的评述。

3.2 仿真软件包与编程语言的比较

在进行仿真研究中，建模人员或分析人员必须做出的最重要的决策之一是选择仿真软件。如果选用的软件不够灵活或者太难用，仿真项目可能产生错误的结果甚至难以完成。使用仿真软件包比使用通用编程语言具有以下优势。

- 仿真软件包自动提供构建一个仿真模型所需要的大部分特征(参见第3.1节至第3.4节)，可以显著缩短“编程”时间并减少整个项目成本。
- 提供了一种仿真建模的自然框架。相比于像C语言这样的通用编程语言，仿真软件包的基本建模构件更切合于仿真。
- 用仿真软件包时生成的仿真模型通常更易于修改和维护。
- 仿真软件包提供更好的故障检测，因为许多潜在类型的错误是自动检查的。因为模型中必须包括的建模构件数量越少，产生错误的机会也越少(不过，用户也很难发现新版本仿真软件包本身的错误，而且有时因为缺乏相关的文档，软件可能用得不对)。

另一方面，许多仿真模型(特别是与国防相关的应用)仍然是使用通用编程语言来编写的，这样的选择的优点如下。

- 大部分建模人员熟悉编程语言，却未必熟悉仿真软件包。

- 与用仿真软件包开发的模型相比，采用 C、C++ 或者 Java 语言编写的高效仿真模型所需执行时间会短一些。这是因为仿真软件包的设计是通过一组建模构件来满足各种各样系统的，而 C 程序可以更有针对性地面向特定的应用。但是，随着并不昂贵的高速 PC 的可用，这种考虑变得不那么重要了。
- 编程语言比某些仿真软件包具有更佳的编程灵活性。
- 编程语言 C++ 和 Java 是面向对象的(见第 3.6 节)，这一点被许多分析人员和程序员所重视，如国防工业的分析人员和程序员。另一方面，大部分仿真软件包并不是真正面向对象的。
- 软件成本通常较低，但整个项目成本未必如此。

虽然使用两种类型的软件都具有各自的优势，我们相信，一般来说，建模者会更慎重地考虑选择使用仿真软件包。如果确实做出了这样的决定，那么，我们觉得，第 3.4 节中讨论的准则对选择特定的仿真软件包来说是有用的。

3.3 仿真软件分类

本节讨论仿真软件包的各个方面。

3.3.1 通用与面向应用的仿真软件包的比较

离散事件仿真软件包主要分为两类，即通用仿真软件包和面向应用的仿真软件包。通用仿真软件包可以用于任何应用，但可以有某些应用的专门特点(比如面向制造和面向过程重组)。另一方面，面向应用的仿真软件包则设计成用于某类应用，例如制造业、医疗保健或者通信网络。第 3.7 节中给出了面向应用的仿真软件包清单。

3.3.2 建模方法

在第 1 章和第 2 章的程序中，我们采用了事件调度的方法进行离散事件仿真建模。系统建模包括识别其特征事件，进而编写一组事件子例程，子例程给出每个事件时刻发生的状态变化的细节。通过按事件发生时间递增顺序执行事件来实现仿真随时间变化的演变。这里，事件子例程的基本性质是在其执行期仿真时钟不推进。

另一方面，大多数当代仿真软件包使用进程方法进行仿真建模。进程是用时间区间来分隔的有时序的互相关联的事件序列，它描述了一个“实体”流经“系统”时的完整经历。实体到达单服务台并得到服务所对应的进程如图 3.1 所示。一个系统或者仿真模型可以有多个不同类型的进程。对应于模型中的每一个进程，都有一个进程“子例程”，它描述其“进程实体”移动通过对应进程的整个历史。进程子例程显式地包括仿真时间推进并且通常配置多个进入点。

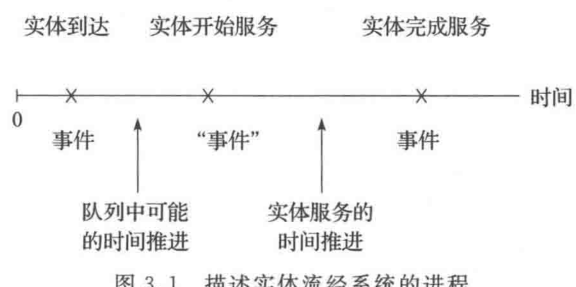


图 3.1 描述实体流经系统的进程

为了更加简洁地说明进程法的性质，图 3.2 给出了单服务台排队系统情形下的标准顾客进程例程的流程图(该进程子例程描述了顾客通过系统的完整经历)。与事件子例程不同，该进程子例程在 1、5、9 程序块有多个进入点。在程序块 1 进入这个子例程对应于顾客户实体的到达事件，这是事件表中最先发生的事件。在程序块 1 中，下一个顾客实体到达的到达事件记录放到事件表中(该下一个顾客实体到达的时间等于当前顾客实体到达的时间加上到达间隔时间)。为了确定当前到达的顾客实体是否能开始服务，在程序块 2 进行检查，看服务台是否空闲。如果服务台忙，将该顾客实体放入队尾(程序块 3)并等待(程序块 4)，直到某个未定的未来时刻被选中为止(称为条件等待)。然后，控制返回到“定时子例程”以确定哪个顾客实体事件是当前最早即将发生的(如果我们把像

图 3.2 那样的流程图设想为系统中的每个顾客实体实际流程，控制随后会转移到流程图中适当的进入点，该进入点对应于某个其他顾客最早即将发生的事件)。当这个顾客实体(安排在程序块 4 等待的实体)在未来的某个点被激活时(此时它在队列的最前面并且另一个顾客完成了服务使得服务台闲)，它在程序块 5 从队列中移出并立即开始接受服务，因而使服务台忙(程序块 6)。到达时发现服务台闲的顾客实体也会立即接受服务(程序块 6)；无论发生哪种情况，我们现在位于程序块 7。在这里将确定开始服务的顾客的离去时间，并在事件表中加入一个对应的事件记录。该顾客实体随后开始等待(程序块 8)直到它的服务完成为止(这是一个无条件等待，因为它的激活时间是已知的)。控制权返回计时子例程，以确定下一个处理的顾客实体是哪一个。在程序块 8 等待的顾客在其服务结束后被激活，这使得在程序块 9 置服务台为闲(允许队列中的第一个顾客立即被激活)，随后在程序块 10 该顾客离开系统。

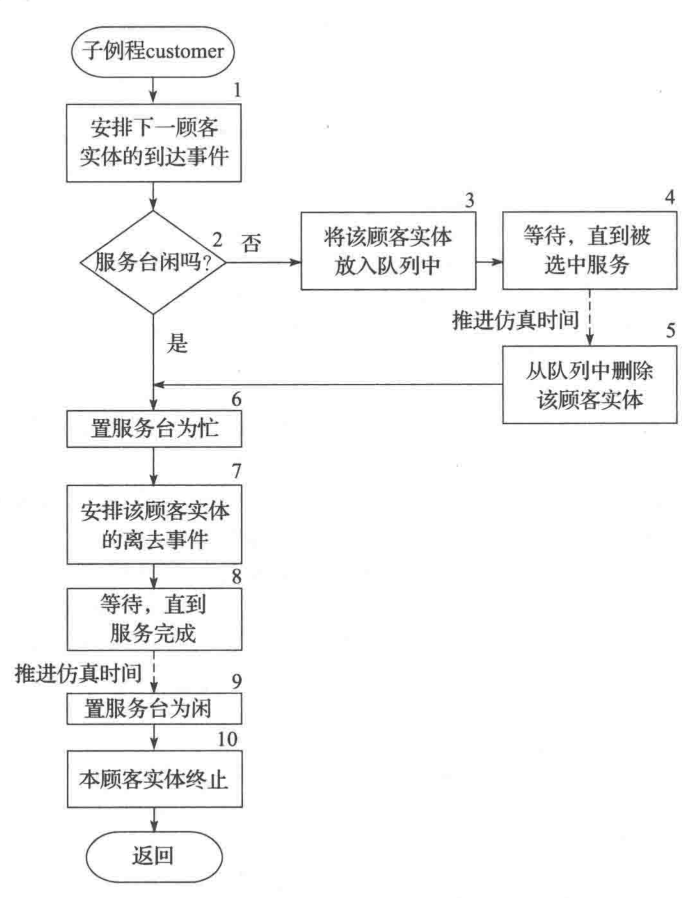


图 3.2 单服务台排队系统的标准顾客进程子例程

进程法也是通过按照事件发生的时间顺序执行事件来实现仿真随时间演变的。从内部来看，进程法和时间调度法对仿真来说是非常相似的(例如，两种方法都用一个仿真钟、一个事件表、一个定时子例程，等等)。然而，进程法在某种意义上更加自然，因为一个进程子例程描述了对应进程实体的完整经历。

3.3.3 通用建模元素

仿真软件包典型元素有实体、属性、资源和队列，它们作为其建模框架一部分。实体(例子参见表 3.1)产生，通过仿真系统的某一部分，随后被销毁。实体通过它们的属性来彼此相区别，属性是与实体存储在一起的一些信息片段。当实体通过仿真系统时，它请求使用资源。如果所请求的资源不可用，则将实体加入队列。在特定队列中的实体也许按照 FIFO

(先进先出)的方式、LIFO(后进先出)的方式，或者按某个属性递增或递减的顺序接受服务。

表 3.1 某些一般仿真应用的实体、属性、资源和队列

系统类型	实体	属性	资源	队列
制造业	零件	零件号，交货期	机器，工人	队列或缓冲区
通讯	报文	目的地，报文长度	节点，链路	缓冲区
机场	飞机	航班号，载重	跑道，门	队列
保险代理	申请人，索赔单	姓名，保险单号码，数额	代理商，接待员	队列

3.4 期望的软件特点

选择仿真软件时有许多特征需要考虑，我们将其归类为以下几组：

- 通用能力(包括建模灵活性和易用性)；
- 软硬件要求；
- 动画；
- 统计特征；
- 用户支持和文档；
- 输出报告和图表。

下面我们依次讨论每组特征。

3.4.1 通用能力

按我们的观点，一个仿真软件产品需要具备的最重要的特征是*建模灵活性*，换句话说，就是对运作流程具有任意复杂性的系统进行建模的能力。注意，完全相同的两个系统是没有的。因此，固定数量的建模构件不具有以任意方式进行某一类编程的能力，对某一类在实际中遇到的系统来说，依赖这类仿真软件包肯定是不合适的。理想情况下，仅使用该软件所提供的构件就应该能够对任意系统建模——不必使用如C这样的编程语言编写的子例程。下面是一些能够增加仿真产品灵活性的特定能力。

- 能定义和改变实体属性以及全局变量，并且在判定逻辑(例如，if-then-else 结构)中使用这两者。
- 能使用数学表达式和数学函数(例如对数函数、指数函数等)。
- 能创建新的建模构件与修改现有的建模构件，以及将他们存储在库中以便当前或未来模型使用。

仿真产品第二个最重要的特征是*易用性*(并且易于学习)，为此当代许多仿真软件包有图形用户界面。仿真产品必须有建模构件(例如，图标或模块)，这些建模构件既不能太“细”也不能太“宏”。前者会导致即使相对简单的情况也需要大量的构件；后者如果为了允许足够的灵活性，会导致每个构件的对话框包含大量的选项。一般来说，对话框中使用标签有助于管理大量的选项。

层次化建模在复杂系统建模时非常有用。*层次化*可以让使用者将多个基本的建模构件组合成一个新的更高层次的构件。这些新构件进而可组合成更高层次的构件。后者可以添加到可用构件库中，并且能够在这个模型或者未来的模型中重用(参见第 3.5.2 小节的例子以及第 3.6 节)。对模型逻辑片的重用能力提高了建模效率。层次化在许多仿真软件包中是一个重要的概念，对包含很多图标和模块的图形化模型来说，为管理“乱屏”，它也是一种有效途径。

软件应该有很好的*调试工具*，例如，交互式调试器。一个强有力的调试器能够允许使用者做以下事情：

- 让单一实体通过模型来观察它是否正确地得到处理；

- 每个特定事件发生(例如，机器停机)的时刻观察模型的状态；
- 设定某些属性或变量的值来“强制”实体进入小概率发生的逻辑路径。

模型快速执行对于某些应用来说是十分重要的，例如大军事模型，以及必须处理大量实体的模型(例如，高速通信网络)。我们用六种仿真产品来对一个简单的制造系统编程并发现，对该模型来说，一个产品比另一个产品快10倍。

人们希望能开发用户友好的“*建模前端*”，以便仿真模型由开发者以外的用户使用。这个能力使开发者创建一个界面，利用该界面非专业用户易于输入诸如平均服务时间或者仿真运行多长这样的模型参数。

大部分的仿真软件供应商提供其软件的*运行期版本*[参见 Banks(1996)]。粗略地说，这种版本允许用户通过使用一个用户友好的“前端”改变模型数据，但是非逻辑的改变。运行期版本的应用包括：

- 允许组织中某个部门的员工运行另一个部门的拥有这个软件的开发版本的员工所开发的模型；
- 向设备供应商和系统集成商出售工具；
- 培训。

注意，运行期许可比标准开发许可通常要便宜或者是免费的。

当前人们相当关心的特征是从其他应用输入数据(或者向其他应用输出数据)的能力(例如，Excel 电子数据表或者数据库)。

传统上，仿真产品为感兴趣的系统提供性能度量(例如吞吐量、系统中平均时间等等)。现在一些产品也包含一个成本模块，它允许对诸如设备、劳动力、原材料、半成品、成品等进行成本分配。

在某些离散事件(如炼钢)仿真中，可能必须有一定的连续仿真能力。我们称这种仿真为*组合式离散-连续仿真*(参见第13.4节)。

有时候，有一组用编程语言编写的复杂逻辑需要集成到一个仿真模型中，因此希望仿真软件包具有调用*外部子例程*的能力。

仿真软件包能够易于以非空且闲的状态进行初始化，这是是很有用的。例如，在制造业系统的仿真中，可能要求将模型初始化为所有的机器为忙而所有缓冲区半满，以便减少模型达到“稳态”需要的时间。

另一个有用的特征是*在一次运行结束时可以保存仿真的状态*，且随后可以很容易地用于重新启动仿真。

最后，在购买仿真软件时，*价格*通常是一个重要考虑的因素。目前，PC 上的仿真软件的价格范围从 \$1 500 到 \$100 000 或者更高。然而，也有必须要考虑的其他方面的费用，例如维护费、升级费，以及可能需要的附加软硬件费用(参见第3.4.2小节)。

3.4.2 软硬件需求

在选择仿真软件时，人们必须考虑该软件可以应用于哪种*计算机平台*。几乎所有的软件都可以在基于 Windows 的 PC 机上使用，某些产品也可用于 UNIX 工作站和苹果机。如果一个仿真软件包可用于多种平台，那么它应该是*跨平台兼容的*。同支持何种操作系统一样，也应该考虑软件运行*所需内存*的数量。如果一个仿真模型可以同时在多核处理器上或连接网络的计算机上独立重复运行是非常值得期待的。

3.4.3 动画和动态图形

内置动画的可用性是越来越多地使用仿真建模的原因之一。在一个*动画*中，当仿真模型随时间变化而演变时，系统的关键要素在屏幕上用图标来代表，并且动态改变位置、颜色和形状。例如，在一个制造系统中，当模型中的叉车改变位置时，代表叉车的图标会相应地改变位置，当模型中的机器改变状态(例如，由闲变到忙)时，代表该机器的图标可能

会变换颜色。

下面是一些动画的应用。

- 向管理者或者其他可能没有认识到(或者不关心)这个模型技术细节的人传达仿真模型(或者仿真本身)的本质。
- 调试仿真计算机程序。
- 说明仿真模型不正确。
- 为改善系统运行程序提供建议(某些事情仅仅观察仿真的数值结果可能不清晰)。
- 培训操作人员。
- 推动项目组之间的交流。

动画有两种基本类型：并发的和后处理的(也称为回放)。在并发动画中，动画是在仿真运行的同时显示的。注意，无论如何，当进行正式运行时，动画通常是被关掉的，因为动画会减慢仿真的执行速度。在后处理动画中，仿真中的状态改变被保存在磁盘文件中，并且在仿真结束后被用来驱动图像。某些仿真软件产品两种类型的动画都有。

我们现在讨论动画的期望特征。首先，仿真软件应该提供默认动画作为建模过程的一部分。因为动画基本上是一种沟通的手段，它应该可以创建高分辨率的图标并将其保存下来为以后重用。软件应该有一个标准图标库。软件应该支持图标流畅的移动；图标不应该“闪烁”和“跳跃”。应该可以控制动画加速和减速，可以放大和缩小并且追拍系统不同部分，避免太大而不能显示在一个屏幕上。某些仿真软件有命名动画视图，因此人们可以构建与仿真系统不同部分对应的视图菜单。动画采用基于矢量的图像(画面通过直线、圆弧和填充来绘制)而不是基于像素的图像(画面通过开启和关闭单个像素点来绘制)，这是可取的。前一类图像允许旋转对象(例如，一个直升机螺旋桨)就像车辆在转过一个转角时保持合适的方向那样。

在运行中，某些具有并发动画的仿真产品允许用户在观察动画时在线停止仿真，改变某些模型参数(例如工作站中的机器数)，然后随时重启仿真。然而，如果系统状态和统计计数器没有重置的话，这有可能发生统计错误。

许多仿真软件包提供三维动画(观看动画的有利位置可以沿三个轴变化)，这对管理演示文稿以及对与垂直空间非常重要的情况是非常重要的。在这些产品中，也有可能为动画观看者提供“从一个实体的背面穿越整个系统”的透视图。

应能导入CAD草图和剪贴画到动画中。

人们往往希望仿真运行时屏幕上显示动态图像和统计数据。动态图像如钟表、刻度盘、水准仪(也许代表一个队列)和动态更新的直方图和时间图(参见第3.4.6小节)。后者的例子是在仿真随时间运行过程中更新某个队列中的人数的图表。

3.4.4 统计能力

如果一个仿真产品不具有良好的统计分析特征，那么它就不可能从仿真研究中获得正确的结果。首先，软件必须有一个好的随机数发生器(见第7章)，即有一个[0，1]区间上均匀分布发生独立观察数据的机制。注意，并非计算机上或者软件产品中的所有的随机数发生器都具有可接受的统计特性。发生器应该至少有100个不同的数据流(最好比100多得多)以分配仿真模型中的不同随机源(例如，到达间隔时间或者服务时间)，这样将使得比较不同的系统设计有更具有统计效力的方式(见第11.2节)。如果各数据流使用默认种子——不依赖计算机内部时钟，仿真软件在不同运行中应该产生相同的结果。另一方面，如果需要的话，用户应该能对每个数据流设定种子。

一般来说，每一个被研究的系统中随机性的数据源在仿真模型中应该用概率分布来表示(见第6章)，而不仅仅是得到的平均值。如果能够找到一个标准的理论分布，对于特定随机源来说这是一个好的模型，那么在模型中应该使用这个分布。至少，以下这些连续分布应该是可用的：指数分布、伽马分布、韦布尔(Weibull)分布、对数正态分布、正态分

布、均匀分布、贝塔分布，以及三角分布。最后一个分布通常作为一个没有系统数据可用时的随机源模型。还要注意的是，在实际的仿真中很少输入随机变量是正态分布的。而且下面这些离散分布也应该是可用的：二项分布、几何分布、负二项分布、泊松(Poisson)分布，以及离散均匀分布。

如果不能找到一个理论分布来很好地表达随机源，那么应该采用一个基于数据的实验(或用户自定义)分布。在这种情况下，随机数用于由被观测系统的数据所构造的分布函数中采样。

应该有(单一)命令用于进行仿真模型的独立重复运行(或多次运行)，这意味着：

- 每次运行采用独立的不同随机数组。
- 每次运行采用相同的初始条件。
- 每次运行重置统计计数器。

注意不同运行的仿真结果是独立的，并且是彼此在概率意义上的复制。这样使得(简单)传统的统计方法可用于不同运行的结果(见第 9 章)。

应该有一个统计上健壮的方法以用于构建均值的置信区间(例如，工厂中零件在系统中的平均时间)。方法应该易于理解并且能够提供好的统计结果。在这点上，我们认为重复运行法(见第 9.4.1 小节和第 9.5.2 小节)肯定是较好的方法。

如果一个人试图去确定系统长期的或者稳态的行为特性，那么一般最好为仿真规定一个预热期，即仿真时间点，此时统计计数器(但不是系统的状态)被重置。理想情况下，仿真软件也应该能够基于试运行来确定预热期的值。目前至少有一个仿真产品采用 Welch 图形化的方法(见第 9.5.1 小节)来规定预热期。

使用重复运行法应该可以构建两个仿真系统(例如，当前系统和建议系统)的均值差的置信区间(见第 10.2 节)。

仿真软件应该允许用户指定对何种性能度量收集输出数据，而不是产生大量的用户不感兴趣的默认输出数据。

至少一个仿真产品允许用户利用其软件来进行统计实验设计，如全因子设计和部分因子设计。当我们开展仿真研究时，我们想知道哪些输入因子(决策变量)对感兴趣的性能度量具有最大的影响。实验设计告诉我们要做哪些仿真实验(运行)使得可以确定每个因子的效果。一些设计还允许我们确定因子之间的相互作用。

现在，对计划购买仿真软件的人们来说一个相当感兴趣的话题是“优化”(见第 12.5 节)。假设存在许多感兴趣的决策变量(输入因子)，每一个都有自己的取值范围(决策变量之间还有可能有线性约束)。另外，有一个需要最大化(或者最小化)的目标函数，目标函数是一个或多个仿真输出随机变量(例如，制造系统的吞吐量)和某些决策变量的函数。“优化器”的目标是以智能方式运行仿真模型(每次运行采用决策变量的某些设置值)并且最终确定能够产生最优解或者接近最优解的一组决策变量。这些优化模块采用启发式的方法，诸如遗传算法、模拟退火算法、神经网络算法、分散搜索算法、禁忌搜索算法等。

3.4.5　客户支持和文档

仿真软件供应商应该定期地提供其软件的公共培训，还应该在客户现场提供客户化培训。好的技术支持对回答如何使用软件以及在软件中发现缺陷是非常重要的。技术支持通常以电话帮助的形式，应该最多在一天内得到响应。

好的文档是使用任何软件产品的必备条件。按我们的意见，应该可以不经过正规训练课程就能够学会一个仿真软件包。一般来说会有一个用户指南或者参考手册，应该有大量的详尽例子可用。现在大部分产品有上下文相关的在线帮助，我们认为这是非常重要的(仅仅在软件中有一个文档的副本是不够的)。许多软件有一个“小例子”库来说明各种建模构件。

应该详尽介绍每种建模构件是如何工作的，特别是，如果它的操作过程很复杂，则更

应如此。例如，如果一个通信网络仿真软件产品提供了一个以太局域网的模块，那么它的逻辑应该仔细描述，所做的任何简化假设都应该相对于所说的标准进行。

仿真软件包有大学版的教材可用是令人十分满意的。

大部分的仿真产品提供一个免费的演示盘，在某些情况下，还可以从供应商的网站下载该软件的工作版，工作版允许开发和运行一些小模型。

供应商发布电子版通信并且召开年度用户会议是有用的。供应商应该定期更新软件(大概地说，一年一到两次)。

3.4.6 输出报告和图表

为了估计性能度量应提供标准报告，为管理说明方便，恐怕也应该能够对报告客户化。既然仿真产品应具有足够的灵活性以便它能计算用户定义的性能度量的估计，它也应该可以把这些估计值输出到用户报告中。对于每一项性能度量(例如，工厂，在系统的时间)，通常会给出平均观测值、最小观测值和最大观测值。如果也给出标准方差(基于一次仿真运行)，则用户应能确认该方法是统计可接受的(如第9.5.3小节中讨论的，具有适当批量大小的批均值法)，否则应视其为非常不可信的[方差和标准差的估计需要独立数据，很少是由仿真模型的一次运行来产生的(见第4.4节)]。应该能够获得在仿真运行期的中间点以及在仿真结束时的报告。

仿真产品应该提供多种(静态)图表。首先，应该可以做出一组观测数据的直方图(见图14.29)。对于连续(离散)数据，直方图是对产生该数据的基本概率密度(质量)函数的图形化估计。时间曲线图也非常重要。在时间曲线图中(例如，参见图14.27)，画出一个或多个关键系统变量(例如，在某些队列中的人数)沿仿真长度的图，提供了被仿真系统的动态行为特性的长期标识(动画提供被仿真系统的动态行为特性的短期标识)。某些仿真产品也允许用柱形图或饼图显示仿真的结果。最后，为度量由一次仿真运行产生的输出数据中的相关性，相关图(参见图6.29)是一种有用的方法。

仿真软件应该能将单个模型的输出观测值(例如在系统中的次数)输出到其他软件包中，例如电子表格、数据库、统计软件包，以及图形软件包等，以便进一步分析和显示。

3.5 通用仿真软件包

在第3.5.1小节和第3.5.2小节中我们分别给出关于Arena和Extend的简要介绍，它们是(在写本书时)两种流行的通用仿真软件包。对于每一个软件我们也展示如何建立一个小工厂的模型。第3.5.4小节列出了另外的一些通用仿真软件包。

3.5.1 Arena软件包

Arena软件包[见Rockwell(2013)和Kelton et al.(2010)]是由罗克韦尔工业自动化公司(韦克斯福德(Wexford)，宾夕法尼亚州)销售的一款通用仿真软件包，它广泛地用于诸如制造业、供应链、国防、医疗保健和呼叫中心。Arena软件包有几个不同的版本，包括基础版、专业版和呼叫中心版。

建模构件在Arena软件包中称为“模块”(模块包括逻辑、用户界面，在某些情况下还包括动画选项)，它们依功能置于许多“模板”中。“Basic Process”模板包括的模块事实上在每个模型中都要用于对实体的到达、离去、服务和决策逻辑的建模。“Advanced Process”模板包括的模块用于执行更高级的过程逻辑和访问在Excel、Access和SQL数据库中的外部数据文件。“Advanced Transfer”模板包括的模块用于对各种类型的传送带、叉式升降装卸车、自动导向车，以及其他物料运输设备建模。“Flow Process”模块用于对油罐、管道、阀门和批处理操作建模。还有，低级别的“Blocks”和“Elements”模板用于某些复杂现实系统建模；这两个模板构成了以前称之为SIMAN的仿真语言。

在Arena中模型构建是通过将模块拖到模型窗口中，将它们连接起来，指明实体通过

被仿真的系统的流动，然后通过对话框或者 Arena 内置电子数据表填写模块细节。模型层次可以没有限制。

Arena 有三维(3D)动画并允许显示动态图形(例如，直方图和时间曲线图)。它还允许人们“观测逻辑执行”并进行细致的图形模型调试。“Arena 三维播放器”是一个可选包，用于创建和观看三维动画。Arena 可以直接生成 .avi 文件以便与其他人分享动画，免费试用版本也是可用的(见第 3.4.1 小节)。

Arena 可以提供无限数量的随机数流(见第 7 章)。此外，用户可以访问 12 个标准的理论概率分布，还可以访问经验分布。Arena 内置对非平稳泊松过程建模的功能(见第 6.12.2 小节)，这是实体到达具有时变速率的模型。

有一个简单的机制用于对特定仿真系统进行独立重复运行并得到感兴趣的性能度量的点估计和置信区间。它也能构建两系统均值差的置信区间。可以获得许多图表，例如直方图、时间曲线图、柱形图表，以及相关图。大部分 Arena 版本都默认包含“OptQuest for Arena”(见第 12.5.2 小节)优化模块。

基于活动的成本核算纳入 Arena 之中，提供增值和非增值的成本及时间的报告。仿真结果存储在数据库中，并使用嵌入 Arena 的 Crystal Reports 给出。

在 Arena 中可以使用 Microsoft Visual Basic for Application(VBA)和复杂的 ActiveX 对象模型。这种能力允许有更加细致的控制和逻辑，包括为输入模型参数建立用户友好的界面和生成定制的报告等等。这项技术也应用于许多外部应用的 Arena 接口，包括 Visio 绘图软件包。

Arena 专业版具有创建定制模块并将其存储在新模板中的能力。Arena 还有一个选项允许模型按实际时间运行并与其他进程动态交互；这支持了如高层体系结构(HLA)(见第 1.6.2 小节)和软/硬件控制系统测试等应用。

这里，我们开发了例 9.25 中的简单的制造系统的 Arena 模型，它由一台机器和一个检查员组成。不过，我们在这里假设机器永远不会发生故障。图 3.3 显示了 5 个需要的逻辑模块和定义实体流所必需的连接。

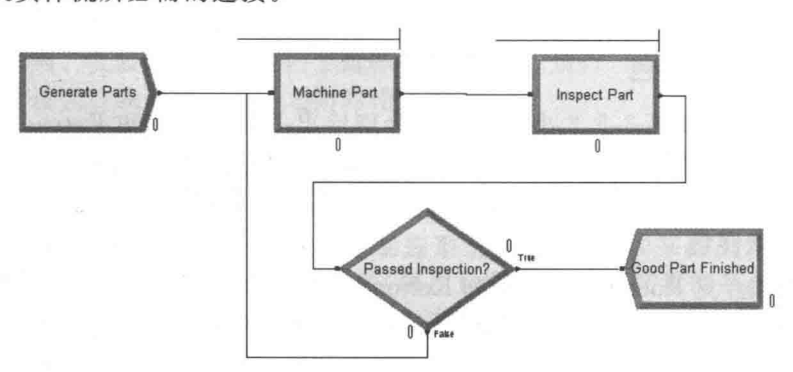


图 3.3 制造系统的 Arena 模型

“Create”模块的对话框如图 3.4 所示，用于生成零件的到达。我们标记该模块为“Generate Parts”，并指定到达间隔时间是均值为 1 分钟的指数分布[记为“Random (Expo)”]。将 Create 模块连接到“Process”模块(见图 3.5)，表示零件在该机器上加工。该“Process”模块标记为“Machine Part”的模块，有一个资源，取名“Machine”，单位为 1，并且其加工时间是 0.65～0.70 分钟之间的均匀

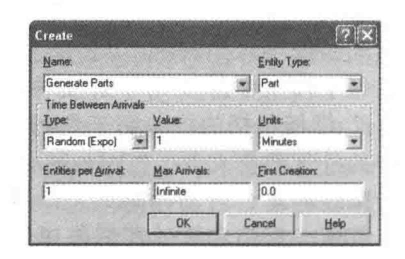


图 3.4 Arena Create 模块“Generate Parts”的会话框

分布。

下一个 Process 模块(见图 3.6)用于表示检查员。我们指定检查时间是 0.75 到 0.8 分钟之间的均匀分布。在检查之后，模块“Decide”(见图 3.7)规定零件可以有一个或者两个结果：“True”(90%的时间会发生)或者“False”。如果零件是好的(True)，那么它被送到标记为“Good Part Finished”的“Depart”模块(未显示)，它在此处消失。否则(False)，它返回到 Machine Part 模块来重新加工。

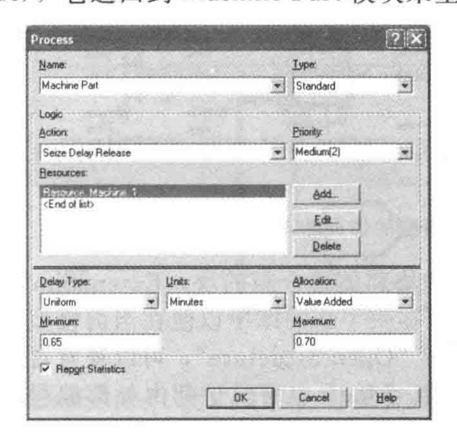


图 3.5 Arena Process 模块“Machine Part”的会话框

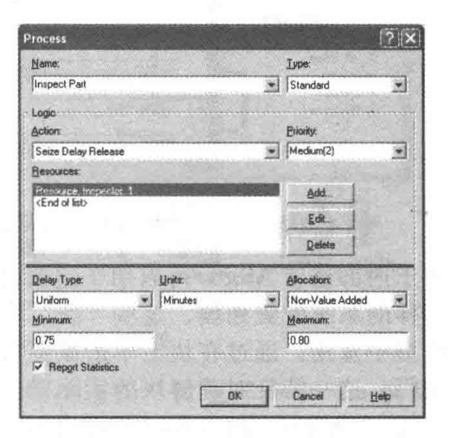


图 3.6 Arena Process 模块“Inspect Part”的会话框

最后，我们需要使用 Run→Setup(见图 3.8)来规定实验参数。我们说一次运行长度要 100 000 分钟。

仿真运行的结果在图 3.9 中给出，我们从中看到一个零件在系统中的平均时间是 4.64 分钟。在屏幕的左手边的选项可以获得其他的输出统计数据。

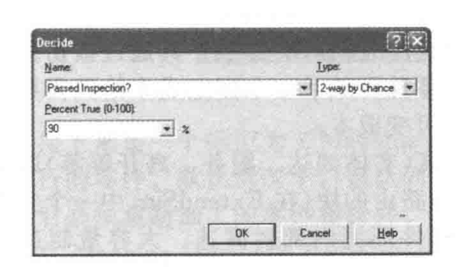


图 3.7 Arena Decide 模块“Passed Inspection”的会话框

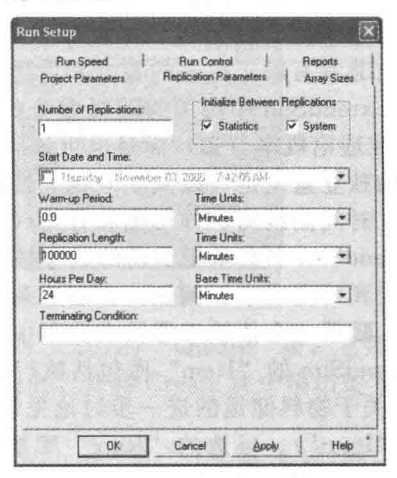


图 3.8 Arena Run Setup 配置选项的会话框

3.5.2 ExtendSim

ExtendSim[见 Imagine(2013)]是 Imagine That，Inc.(圣·何塞(San Jose)，加利福尼亚州)公司销售的 4 个通用仿真软件包的系列名称。每个 ExtendSim 产品有针对特定市场的组件，但是所有的产品共享一套核心功能。模型的构建是通过从库(项、值、绘图器等)中选择块，在模型窗口的合适位置放置块，将块连接起来表示系统中的实体(或者值)的流动，随后使用对话框给出块的细节。

	Average	Half Width	Minimum Value	Maximum Value
Wait Time				
Part	3.0280	0.247628038	0.00	94.4532
Transfer Time				
Part	0.00	0.000000000	0.00	0.00
Other Time				
Part	0.00	0.000000000	0.00	0.00
Total Time				
Part	4.6397	0.248893749	1.4001	101.70

图 3.9 制造系统 Arena 模型的仿真结果

由于内部语言 ModL 可以用于定制已经存在的块和创建全新的块，ExtendSim 可以为多种多样的系统配置建模。这些“新”块可以放置在一个新的库中以便在当前模型或者未来的模型中重用。通过在块上单击鼠标右键并选择“Open Structure”，可以查看对应特定块的代码；这一特征对理解块的实际操作非常有用。ModL 也可以访问由外部编程语言如 Visual Basic 和 C++ 语言创建的应用和过程。

模型的层数可以没有限制（见下文）且也可以使用继承（见第 3.6 节）。一个“Navigator”允许我们从一个层次递阶移动到另一个层次。所有的 ExtendSim 产品提供基础的二维动画，而 ExtendSim 套装产品还提供三维动画。Proof Animation[见 Wolverine (2006)]作为可选项提供给用户选用。

在 ExtendSim 中的每个仿真模型有一个关联的“Notebook”，它可以包括图片、文本、对话项和模型结果。因此，一个 Notebook 可以用作模型的“前端”或者用作仿真实际运行时显示重要模型结果的工具。模型的内部关系数据库中也能存储并访问每个模型的参数；这非常有助于数据的整合和管理。

在 ExtendSim 中可用的随机数流的数目没有本质上的限制。此外，用户还可以访问 34 个标准理论概率分布，而且也可访问经验分布。ExtendSim 有一个简单的机制来实现仿真模型的独立重复运行以及获取相关性能度量的点估计和置信区间。有许多图表可用，如直方图、时间曲线图、柱形图和甘特(Gantt)图。

ExtendSim 中有基于活动的成本核算能力，它允许人们对实体赋予其通过被仿真系统时的固定和可变成本。例如，在制造系统中，一个零件可赋予一个固定成本给其需要的原始材料，赋予一个依赖于零件在队列中等待时长的可变成本。

ExtendSim 的“Item”库包括执行离散事件仿真（实体到达、服务、离开等等）以及物料储运（关于物料储运的进一步讨论见第 14.3 节）与路由的块（在 ExtendSim 中一个实体称为一个“Item”）。可选的“Rate”库提供在离散事件环境内部对高速、大容量制造系统（例如罐装生产线）建模的块。在“Value”库中的块用于执行连续仿真（见第 13.3 节）并为离散事件仿真提供建模支持（数学计算、基于仿真的优化，以及与其他应用共享数据等）。

ExtendSim 软件的“场景管理器”(Scenario Manager)（允许建模者研究仿真模型如何响应从一个场景（模型输入参数或因子的一组值）到另一个场景变化，在工厂设计过程中感兴趣的场景可以用场景管理器手工输入或者自动描述（参见第 12.2 节）。此外，建模者可以指定期望的描述场景独立的运行（每次用不同的随机数列）次数。场景管理器迭代地运行场景，记录每一个场景运行的响应，而且通过每一个场景运行来总结响应。模型因素和它相应的响应可以导出到 ExtendSim 软件的数据库、JMP 和 Minitab 统计包，或者导出到 Excel 供进一步分析使用。ExtendSim 软件的也有优化的模块（参见第 12.5.2 小节）。

我们现在说明如何建立在第3.5.1小节中讨论的制造系统的ExtendSim模型。特别地，图3.10显示了该模型需要的块和连接；连接对应实体(这个系统的零件)的流动。我们已经在每个块下面放置了一个描述性的标签，在下面该模型的讨论中我们援引这些标签(注意，在这里展示的块和在Extend的最终发布的版本7中的外观上可能有一些小差别)。

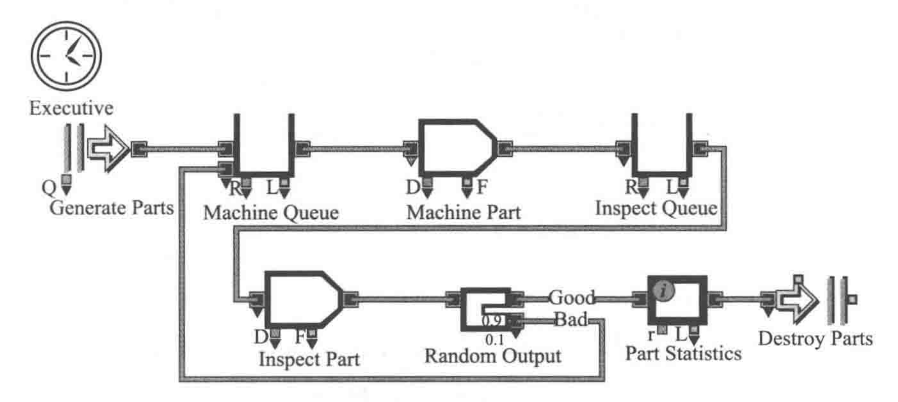


图3.10 制造系统的ExtendSim模型

“Executive”块，图形上它没有与任何其他块相连接，管理ExtendSim模型的事件表。模型中实际上的第一个块是一个标记为“Generate Parts”(它的对话框见图3.11)的“Create”块，用来产生均值为1分钟的指数到达间隔时间的零件。

图3.11 ExtendSim Create块“Generate Parts”的对话框

接下来是一个标记为“Machine Queue”(见图3.12)的“Queue”块，它用来存储等待执行的零件。默认情况下，该队列的容量是无限的，并将Create块中输出的零件同经过检验后必须重新加工的零件相合并。

在Machine Queue块之后的是一个标记为“Machine Part”(见图3.12)的“Activity”块。如对话框中所示(见图3.13)，我们指定在一个时间只能加工一个零件。我们还选择“Uniform，Real”作为加工时间分布，随后分别设定其最小值和最大值分别为0.65分钟和0.70分钟。该Activity块连接到标记为“Inspect Part”的第二个Queue块，这里检查时间为0.75到0.80分钟间的均匀分布。

对应检查员的Workstation块连接到选择Select Output块，它用于确定零件好还是坏。在其对话框中(见图3.14)，我们规定零件随机地离开块的输出端。在表格中我们输入概率值0.9和0.1，表明90%的零件为“好”通过上面的输出端，10%的零件为“坏”通过下面的输出端。我们也可以选择在这个块的输出连接处显示的概率。

图 3.12 ExtendSim Queue 块“Machine Queue”的对话框

图 3.13 ExtendSim Activity 块“Machine Part”的对话框

模型中的下一个块是一个标记为“Part Statistics”的“Item Information”块，计算完成零件的输出统计。在其对话框中(见图 3.15)，我们看到 100 078 个(好)零件已完成，在系统中平均时间(周期时间)是 4.46 分钟。模型中的最后一个块是一个标记为“Destroy Parts”(见图 3.10)的“Exit”块，在这里完成的零件从模型中删除。

模型的时间单位(分钟)，仿真运行长度(100 000)，期望运行次数(1)在“Simulation Setup”选项中规定，该选项从屏幕最上端“Run”下拉菜单(没有显示)来访问。模型的 Notebook(见图 3.16)可以从“Window”下拉菜单访问，它将模型重要的输入参数和结果集合在一起。

在图 3.17 所示模型中，我们给出了采用层次(见第 3.4.1 小节)的 ExtendSim 模型的一个版本。如果我们在名为“Process”的分层块上双击(在层次结构中的第一层)，然后我们进入了层次结构的第二层，在这里我们看到原始的 Machine Queue 和 Machine Part 块，如图 3.18 所示。

图 3.14 ExtendSim Select Item Out 块“Random Output”的对话框

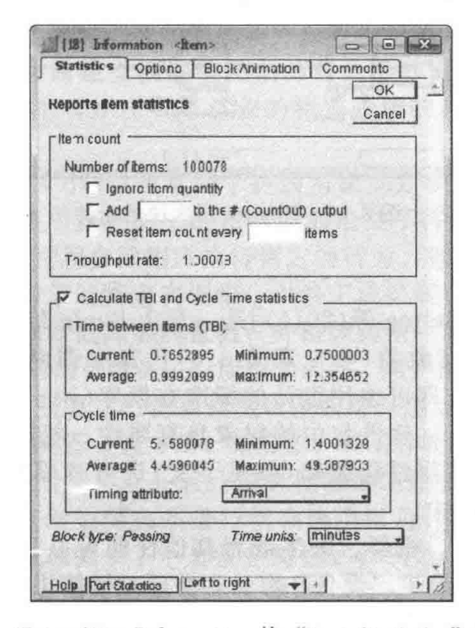


图 3.15 ExtendSim Information 块“Part Statistics”的对话框

图 3.16 制造系统的 ExtendSim Notebook

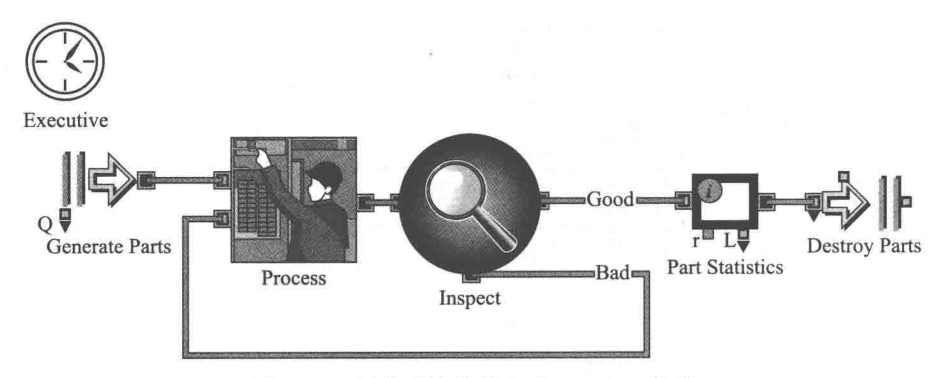


图 3.17 制造系统的层次 ExtendSim 模型

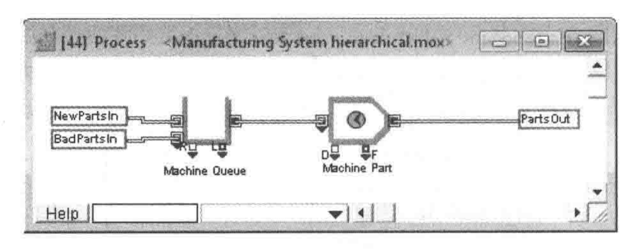


图 3.18 Process 层次块的组件

3.5.3 Simio

Simio[Simio(2013)和 Kelton 等(2011)]是一套由 Simio LLC(塞威克利(Sewichkley),宾夕法尼亚州)经营的面向对象的(参见章节 3.6)仿真和调度产品。Simio 软件包是基于智能对象的仿真建模架构，即可使用默认的标准数据库(用于离散事件仿真)，也可以图形方式创建全新的对象(Simio 软件包中的对象具有属性、状态和逻辑性)。包含 15 种对象定义的标准数据库可以使用过程逻辑(参见下文)进行建模和扩充，使用其他的仿真项目可以将新的对象存储在库中。

库中的对象可能是客户、机器、医生或者其他任何可以在系统中找到的对象。Simio 软件包的建模过程是将对象拖进“设备”窗口，并通过链接将其进行连接以表示仿真系统中实体的流动，再利用属性编辑器对对象进行详细描述。模型逻辑性和动画在单个步骤中建立，为了便于建模通常在二维视图中进行。但是仅需一个按键便可转换为三维全景视图。

在 Simio 软件包中建立一个对象等同于建立一个模型，因为这两种结构并无差别。每当建立了一个模型，也同时定义了一个能够应用于另一个模型的对象。例如，如果在一个工作站模型中结合了两台机器和一个机器人，则工作站模型本身就是一个对象，并且可以被另一个模型所使用。Simio 软件包中建立的每一个模型都是一个可以用于构建层次模型的自动生成块。

当实例化一个对象转换为模型时，可以指定对象的“属性”(静态输入参数)，以管理指定对象实例的行为。例如，一台机器的属性可能是其加工时间。对象的开发人员决定了属性的数量及其含义。Simio 软件包中的属性可以是数值型、布尔型，以及字符串等。

除了属性之外，对象还具有“状态”，该状态可随着对象逻辑性的执行而更改其设定值。一台机器的状态可能是其工作状态(例如，繁忙或者空闲)。属性(propority)和状态共同组成了一个对象的属性(attribute)。

Simio 软件包中的对象可以根据 5 个基础类中的一个类进行定义，这 5 个基础类为对象提供了基础行为。第一个类为“固定对象”，在模型中具有一个固定位置，用于代表系

统中不会从一个位置移动到另一个位置的某事物，例如工厂中的一台机器或者医院里的一个手术室。

“实体”是一个可以在三维空间中的链接网络和节点上移动的对象。实体可以是制造加工系统中的零件或者是医院里的病人、护士和医生。注意，在传统仿真包中实体是被动的并受模型过程的支配(参见第3.3.2小节)。但是Simio软件包中的实体为可以控制自身行为的智能对象。

“链接”和“节点”对象用于建立实体可以流动的网络。链接为实体定义了一条从一个对象移动到另一个对象的路径，而节点定义了链接的起点和终点。链接和节点可以共同结成网络。链接可以是一个带有固定行程时间的电梯，也可以代表一次输送。

对象的最后一个类是一个“运输器”，是实体类的一个子类。运输器是一个实体，具有装载、运输，以及卸载一个或多个实体的附加性能。运输器可以用于建立公共汽车、叉车，以及任何能够将实体从一个位置移动到另一个位置的对象的建模。

Simio软件包的关键特征是具有从一个基础类中创建多种对象行为的能力。Simio建模框架是应用领域中性的，例如，这些基础类并没有指定给某个特定的如制造业或者医疗等应用领域。但是，却很容易建立面向应用的库，这些库是由基础类中的智能对象组成的。Simio软件包的设计理念显示出属于对象的特定领域的逻辑是由用户建立的，并没有编程到核心系统中去。

过程方法(参见第3.3.2小节)通常用于扩展对象的逻辑性或者建立新的对象。Simio中利用流程图定义过程，其中流程图中的每一步都定义了待执行的某个行为。Simio中有超过50个可用的、不同的过程步骤用来执行指定的行为，如时间延迟、等待抢占资源等。过程逻辑可以插入到一个对象的特定实例中，以修正或扩展自身行为。例如，一个代表机器的对象可以使用过程逻辑在故障期间抢占，并持续占用一名修理工人。

Simio软件包中有无限的随机数流可以应用。此外，用户能够访问19种标准理论概论分布和经验分布。有一种独立重复运行仿真模型以及获得兴趣interest性能度量点估计和区间估计的简单机制。有包括时间图、直方图、条形图、饼图等在内的大量可应用的图表。

Simio语言提供了一个用于建立并运行仿真模型的三维交互环境，对仿真项目的利益相关这来说，该环境有益于对空间关系进行精确建模以及计算模型行为。但是，Simio语言同样对执行和分析仿真试验提供了一套复杂特性。特别是，模型可以具有指定执行一套方案的相关“经验”。每个方案可以具有一个或多个输入控制并将具有一个或多个输出响应。输入控制为一个方案变化到另一个方案的因子(例如，工作站的机器数量)，输出响应为用来估计不同方案效能的性能测度(例如，零件在系统内的平均时间)，此外，每种方案可以重复指定次数，这些重复可以通过多个处理器核心或者网络中不同的计算机来同时执行，此举将大大减少试验所需的时间。Simio语言的嵌入式分析工具包括一个从一套候选方案中自动选择最佳方案的程序[参见第10.4.3小节以及Kim和Nelson(2001)]和SMORE图[参见Nelson(2008)]。SMORE图同时显示了响应期望值的点估计和区间估计，以及一个附加的盒图[参见第6.4.3小节]。“Simio的最优化搜寻”[参见第12.5.2小节]优化模块作为一个选项是可用的。

尽管Simio语言主要是利用面向对象方法面向执行离散事件仿真的，但是Simio语言同样支持连续流的系统建模，执行基于智能体(Agent)仿真(因为它的面向对象性)，以及利用过程方法执行离散事件仿真。此外，Simio语言同样可以用于在实际执行过程中作为基于风险的规划，以及提高组织日常运作的调度工具。

我们现在开发第3.5.1小节中讨论的简单制造系统的Simio模型。图3.19给出了该系统的Simio模型，该模型由如下对象组成：名为“Part_Arrivals”的“资源”对象，其功能为用于创建到达系统的工作；名为“Machine_Part”的“服务”对象，其功能是建立机械加工模型；名为“Inspect_Part”的服务对象，其中，实体离开了系统。在本例

中，我们使用一个名为“Connector”的零时链接来定义资源、服务和接受对象之间的传播路径。

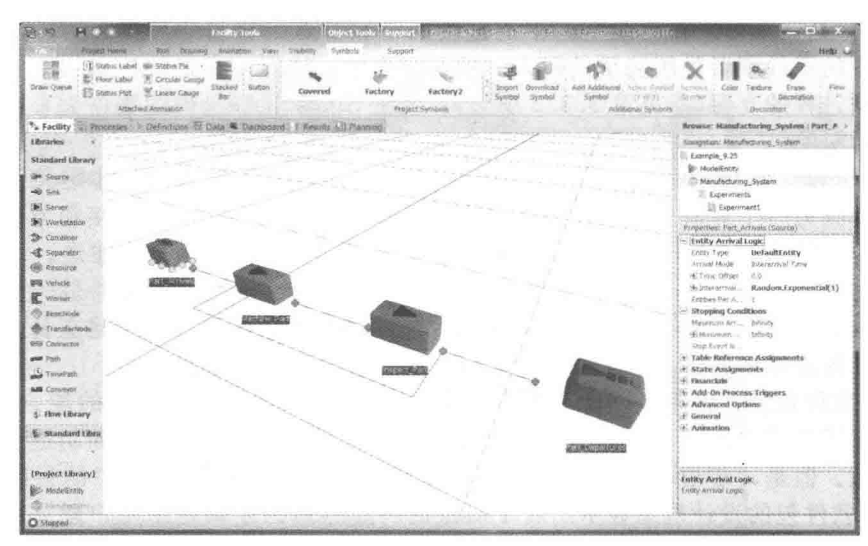


图 3.19 加工制造系统的 Simio 模型

在 Part_Arrivals 对象周围的小圆形“选定句柄”表示该对象是用于编辑的(通过单击)。被选中对象的属性可以通过屏幕右下方的“Property Editor”按钮进行编辑。“Part_Arrivals”对象基于“Arrival Mode”生成系统到达。默认情况下，本例中“Interarrival Time”模型的间隔时间服从均值为 1 分钟的指数分布(另外，“Time var ying Arrival Rate”模型根据非稳态泊松过程生成到达，“Arrive Table”模型使用存储在表格或者例如电子表格等外部数据源中的数据对到达进行调度)。

图 3.20 中显示了 Machine_Part 对象的属性。属性分成不同的类别，这些类别可以通过类别名称左边的+/－符号进行展开和折叠。这些属性指定“Processing Time”在区间[0.65，0.70]分钟上服从均匀分布。注意，该表达式可以直接键入也可以通过字段右侧的下拉箭头(未标示)选定“Expression Editor”进行键入。如果期望“Machine_Part”失效，则可以在“Reliability Logic”目录下方进行选定。“Ginancials”目录可用于指定作业成本的使用率。

我们并未给出 Inspect_Part 对象的属性编辑器，其中 Inspect_Part 对象的间隔时间在区间[0.75，0.80]分钟上服从均匀分布。离开 Inspect_Part 的两个连接器(参见图 3.19)的链接权重分别为 0.9 和 0.1，并对基于“ByLink Weight”的退出节点设定了路由规则。

针对该模型，我们进行了 30 次重复仿真实验并观察一个零件在系统内运行 100 000 分钟的平均时间，图 3.21 给出了该简单实验的规范和部分结果。注意，系统中超过 30 次重复仿真所得到的平均时间为 4.55 分钟。

Properties: Machine_Part (Server)

⊟ Process Logic	
Capacity Type	Fixed
Initial Capacity	1
Ranking Rule	First In First Out
Dynamic Selection ...	None
⊞ Transfer-In Time	0.0
⊟ Processing Time	Random.Uniform(0.65,0.70)
Units	Minutes
⊞ Buffer Capacity	
⊞ Reliability Logic	
⊞ State Assignments	
⊞ Secondary Resources	
⊞ Financials	
⊞ Add-On Process Triggers	
⊞ Advanced Options	
⊞ General	
⊞ Animation	

Process Logic
Process Logic

图 3.20 Machine_Part 对象的属性

图 3.22 以 SMORE 图的形式给出了这些相同的结果。该图显示了一个系统期望平均时间的点估计(“小圆点”)和 95%的置信区间(覆盖在小圆点上面的“小”的矩形阴影)。叠加在此之上的是一个盒图，显示了 30 次观察实验中系统平均时间的最小值、四分之一值、中间值、四分之三值，以及最大值。最后，两个外部阴影矩形为位于第 25 个百分点和第 75 个百分点的 95%置信区间。

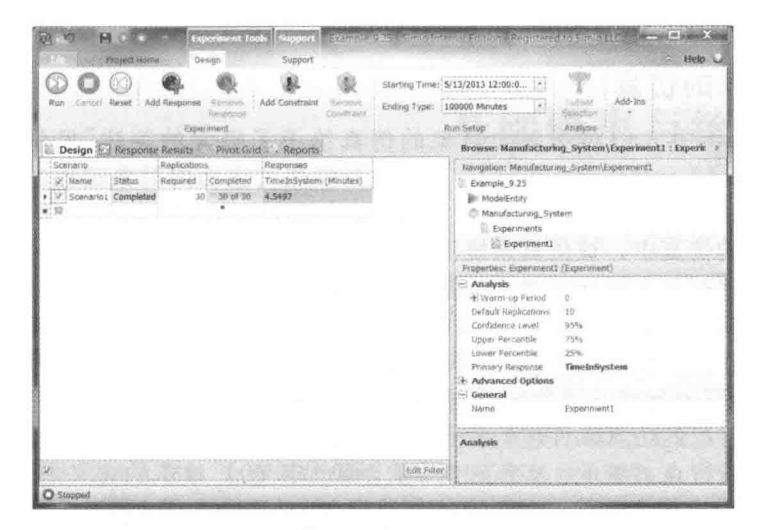


图 3.21 指定一次实验的设计视图

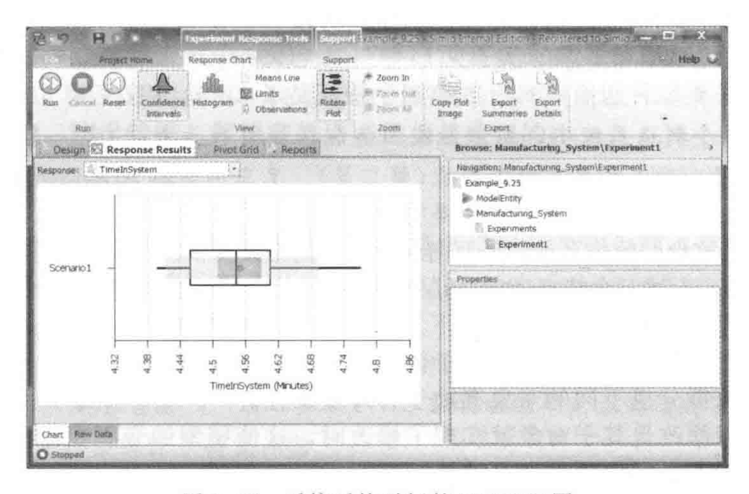


图 3.22 系统平均时间的 SMORE 图

仿真模型生成的标准报告可以潜在地包含大量的输出统计数据，使其很难找到感兴趣的信息。为了缓解这一难题，Simio 将仿真结果以“Pivot Grid”(类似于 Excel 中的透视表)形式呈现，此举可以很容易以适当的形式定制显示感兴趣的统计数据。图 3.23 给出了以“透视表格”形式显示的仿真结果，该结果由 30 次重复实验生成。注意，机器和检查员的使用率分别为 0.75 和 0.86。

```
void report(void)  /* Report generator function. */
{
    /* Get and write out estimates of desired measures of performance. */

    fprintf(outfile, "\n\n%5d%16.3f%16.3f%16.3f", num_terms,
            sampst(0.0, -SAMPST_RESPONSE_TIMES), filest(LIST_QUEUE),
            filest(LIST_CPU));
}
```

图 3.23 “透视表格”形式显示的仿真结果

3.5.4 其他通用仿真软件包

还有几种其他著名的通用仿真软件包，包括 AnyLogic[AnyLogic(2013)]，SIMUL8

[SIMUL8(2013)]以及 SLX[Wolverine(2013)]。

3.6 面向对象的仿真

在过去的 20 年中人们对于面向对象的仿真给予了极大的关注[见例子，Joines and Roberts(1998) and Levasseur(1996)]。这恐怕是强烈关注面向对象编程的结果。事实上，面向对象的仿真和编程都起源于在 20 世纪 60 年代提出的面向对象的仿真语言 SIMULA。

在面向对象的仿真中，被仿真系统认为是由对象(如实体或者服务台)组成的，当仿真随时间推进时对象彼此之间交互效用。某些对象类(如实体)可能有若干实例在仿真执行期中并发出现。对象包含数据并且有方法(见例 3.1)。数据描述了对象在特定时刻的状态，而方法描述对象能够执行的动作。特定对象实例的数据只能由其自己的方法中改变，其他对象实例(具有相同类型或不同类型的)只能查看它的数据，这称为封装。

真正的面向对象的仿真软件包的例子是 Flexsim 和 SIMSCRIPT III。这样的仿真软件包具有三个主要的特点：继承、多态和封装(上面已定义)。继承的含义是如果以已经存在的对象类型(父亲)的方式定义一个新的对象类型(有时称为儿子)，那么儿子类型“继承”父亲类型的所有特征。可以选择性地改变儿子的某些特征或者添加新的特征。多态是当不同的对象类型有相同的祖先时可以有相同名字的方法，但是当被引用时在不同的对象中可能引起不同的行为[继承和多态的例子见 Levasseur(1996)]。

面向对象的仿真软件包有时具有的第四个特征是*层次化*，定义见第 3.4.1 小节。

例 3.1 在一个制造系统中，制造区域和装配区域可视为对象(第一层)。依次地，制造区域可能由机器、工人和叉车对象组成(第二层)。叉车的数据可能包括它的速度和它能举起的最大重量。一个叉车的方法可能是它用来选择下一个作业的发派规则。 ◀

某些供应商宣称他们的仿真软件是面向对象的，但是某些情况下软件可能不包括继承、多态或者封装。此外，上述三个特点(也包括层次化)中的某些特征有时候被赋予了不同的含义。

面向对象的仿真可能具有的优点如下。

- 由于现有的对象可以重用或者易于修改，因此它提高了代码可重用性。
- 通过把系统拆分成不同的对象有助于管理复杂性。
- 当父对象能修改且其子对象就实现了修改时，这使模型的变更变得容易。
- 使多个编程人员的大项目变得容易。

面向对象的仿真可能的缺点如下。

- 某些面向对象的仿真软件包可能越深入学习越难。
- 必须做许多项目并重用对象才能够充分发挥其优势。

3.7 面向应用的仿真软件包举例

在这节中我们列出一些现在可以获得的面向应用的仿真软件包。

- 制造的仿真软件包：AutoMod[Applied Materials(2013)]，Enterprise Dynamics [INCONTROL(2013)]，FlexSim[FlexSim(2013)]，Plant Simulation [Siemens(2013)]，ProModel[ProModel(2013)]，and WITNESS[Lanner(2013)](进一步的讨论见第 14.3 节)。
- 通信网络的仿真软件包：OPNET Modeler[Riverbed(2013)] and QualNet [SCALABLE(2013)]。
- 医疗保健的仿真软件包：FlexSim Healthcare[FlexSim(2013)]和 MedModel[MedModel (2013)]。
- 流程重组和服务的仿真软件包：Process Simulator[ProModel(2013)]，ProcessModel[ProcessModel(2013)]，ServiceModel[PROMODEL(2005c)]，SIMPROCESS[CACI(2005)]。
- 动画(独立的)的仿真软件包：Proof Animation[Wolverine(2013)]。

第4章 基础概率与统计回顾

4.1 引言

要完成一个成功的仿真研究，不仅仅包括建立待研究系统的流程图，将“流程图”转换成计算机“程序”，然后为每个建议的系统配置进行一次或几次重复运行。使用概率和统计也是构成整个仿真研究必需的组成部分，每个仿真建模小组应该至少包括一个在这方面经过彻底的技术培训的人员。为理解如何构建一个概率系统(见第4.7节)、验证仿真模型(第5章)、选择输入概率分布(第6章)、从这些分布中生成随机样本(第7、8章)、对仿真输出数据进行统计分析(第9、10章)，以及设计仿真实验(第11、12章)，概率统计是特别需要的。

本章中我们会建立贯穿本书中使用的统计符号，并且回顾一下基础的、特别是有关仿真的概率统计知识。我们还要指出，将基于独立观测的经典统计技术应用到仿真输出数据所存在的潜在危险，输出数据，即使有的话，很少是独立的。

4.2 随机变量及其性质

实验是一个结果不确知的过程。实验所有可能结果的集合称为样本空间，记为 S。这些结果本身称为样本空间中的样本点。

例 4.1 如果实验由掷硬币构成，则

$$S = \{H, T\}$$

其中，符号{}表示“集合包含”；“H”和“T”分别表示结果是正面还是背面。◀

例 4.2 如果实验由掷骰子构成，则

$$S = \{1, 2, \cdots, 6\}$$

其中，i 表示结果，即出现在骰子上的数字，$i=1$，2，…，6。◀

随机变量是一个函数(或者规则)，它将一个实数(大于 $-\infty$，而小于 $+\infty$ 的任何数)赋给样本空间 S 中的每一点。

例 4.3 考虑掷一对骰子的实验，则

$$S = \{(1,1), (1,2), \cdots, (6,6)\}$$

其中，(i, j)表示 i 和 j 是分别出现在第一个和第二个骰子上的点数。

如果 X 是对应于两个骰子之和的随机变量，则 X 将值7赋给结果(4，3)。◀

例 4.4 考虑掷两枚硬币的实验，如果 X 是对应于出现正面数的随机变量，则 X 将值1赋给结果(H，T)或者(T，H)。◀

通常，我们用大写字母如 X，Y，Z 标记随机变量，用小写字母如 x，y，z 标记随机变量的取值。

随机变量 X 的分布函数(有时称累积分布函数)$F(x)$定义如下，对于每个实数 x，有

$$F(x) = P(X \leqslant x), -\infty \leqslant x \leqslant +\infty$$

其中，$P(X \leqslant x)$表示事件$\{X \leqslant x\}$的概率[有关事件与概率的讨论见 Ross(2003，第1章)]。因此，$F(x)$是当实验完成时随机变量 X 取值不大于 x 的概率。

分布函数 $F(x)$具有如下性质。

(1) 对于所有的 x，$0\leqslant F(x)\leqslant 1$。

(2) $F(x)$是非递减的[即，如果 $x_1<x_2$，则 $F(x_1)\leqslant F(x_2)$]。

(3) $\lim\limits_{x\to+\infty} F(x)=1$，且 $\lim\limits_{x\to-\infty} F(x)=0$(由于 X 只取有限值)。

随机变量 X 称为是离散的，如果它至多取可数个值，比如说，x_1，x_2，…(“可数”的含义是一组可能的取值可以与一组正整数一一对应。一个不可数集的例子是 0，1 之间的所有实数)。因此，只取有限个数值 x_1，x_2，…，x_n 的随机变量是离散的。离散随机变量 X 取值 x_i 的概率为：

$$p(x_i)=P(X=x_i),\quad i=1,2,\cdots$$

并且必须有：

$$\sum_{i=1}^{+\infty} p(x_i)=1$$

其中，求和的含义是将 $p(x_1)$，$p(x_2)$，…加在一起。

关于 X 的所有概率声明可以从 $p(x)$计算得到(至少在原理上)，$p(x)$称为离散随机变量 X 的概率质量函数。如果 $I=[a,b]$，其中，a 和 b 是满足 $a\leqslant b$ 的实数，则

$$P(X\in I)=\sum_{a\leqslant x_i\leqslant b} p(x_i)$$

其中，符号$\in$表示“属于”，求和表示将所有满足 $a\leqslant x_i\leqslant b$ 的 x_i 对应的 $p(x_i)$加在一起。离散随机变量 X 的分布函数 $F(x)$为：

$$F(x)=\sum_{x_i\leqslant x} p(x_i),\quad 对于所有 -\infty<x<+\infty$$

例 4.5 对第 1.5 节中的库存例子，产品的需求量是一个离散随机变量 X，取值 1，2，3，4 分别对应的概率是$\frac{1}{6}$，$\frac{1}{3}$，$\frac{1}{3}$，$\frac{1}{6}$。X 的概率质量函数和分布函数如图 4.1 和图 4.2 所示。此外，

$$P(2\leqslant X\leqslant 3)=p(2)+p(3)=\frac{1}{3}+\frac{1}{3}=\frac{2}{3}$$

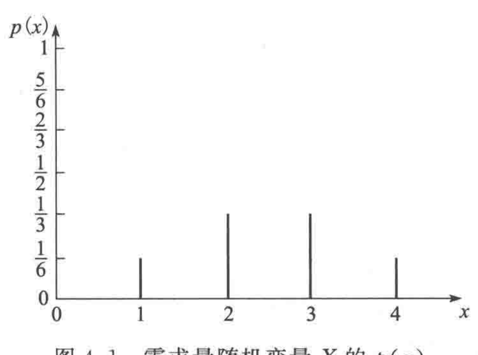


图 4.1　需求量随机变量 X 的 $p(x)$

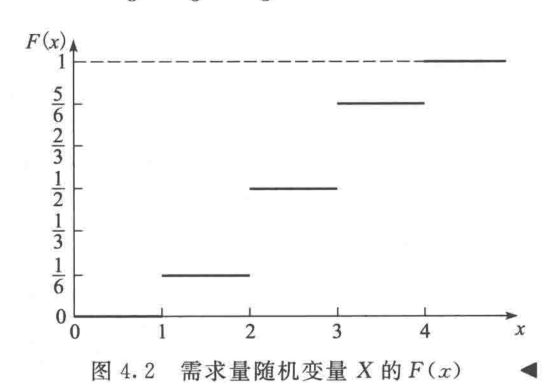


图 4.2　需求量随机变量 X 的 $F(x)$　◀

例 4.6 一个制造系统生产零件随后必须进行零件质量检查。假设 90%被检查的零件是好的(标记为 1)，10%的零件是坏的必须废弃(标记为 0)。如果 X 表示检查一个零件的结果，则 X 是一个离散随机变量 $p(0)=0.1$，$p(1)=0.9$(关于伯努利(Bernoulli)随机变量的讨论见第 6.2.3 小节)。◀

我们现在考虑随机变量可以取不可数的无限个数的不同值(例如，所有非负实数)。称随机变量 X 是连续的，如果存在一个非负函数 $f(x)$，对于任意的实数集 B(例如，B 可以是 1，2 之间的所有实数)，满足

$$P(X\in B)=\int_B f(x)\mathrm{d}x\quad 且\quad \int_{-\infty}^{+\infty} f(x)\mathrm{d}x=1$$

因此，$f(x)$曲线下方的总面积为 1。同样，如果 X 是一个非负随机变量，就像其在仿真应用中的通常情况那样，第二个积分的范围是从 0 到 $+\infty$。所有关于 X 的概率声明可以(原理上)从 $f(x)$计算得到，$f(x)$称为连续随机变量 X 的概率密度函数。

对离散随机变量 X 来说，$p(x)$是值 x 实际概率。然而，$f(x)$不是连续随机变量 X 等于 x 的概率。对于任意实数 x，有：

$$P(X=x)=P(X\in[x,x])=\int_x^x f(y)\mathrm{d}y=0$$

由于对应于每个值 x 的概率都为 0，我们现在给出 $f(x)$的解释。如果 x 是任意数并且 $\Delta x>0$，则有：

$$P(X\in[x,x+\Delta x])=\int_x^{x+\Delta x} f(y)\mathrm{d}y$$

表示 x 和 $x+\Delta x$ 之间的 $f(x)$曲线下方面积，如图 4.3 所示。因此一个连续随机变量 X 在同样宽度的情况下更可能落在 $f(x)$值相对较大的区间中。

连续随机变量 X 的分布函数 $F(x)$为：

$$F(x)=P(X\in(-\infty,x])=\int_{-\infty}^x f(y)\mathrm{d}y,\quad \text{对于所有}-\infty<x<+\infty$$

因此(在某些适当的技术假设下)，$f(x)=F'(x)$[$F(x)$的导数]。进一步，对于任何满足$a<b$的实数 a 和 b，如果 $I=[a, b]$，则有：

$$P(X\in I)=\int_a^b f(y)\mathrm{d}y=F(b)-F(a)$$

其中，最后一个等式是应用了微积分基本定理，因为 $F'(x)=f(x)$。

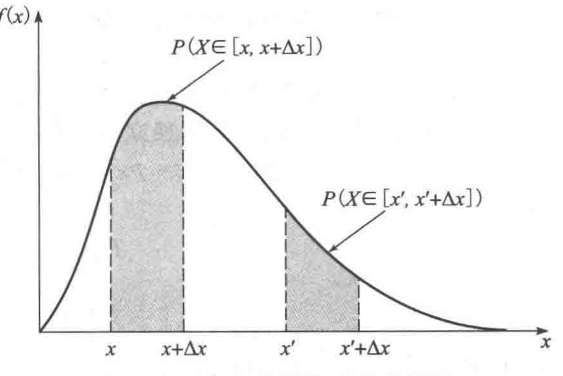


图 4.3　概率密度函数 $f(x)$的解释

例 4.7 [0, 1]区间上的均匀随机变量具有如下概率密度函数

$$f(x)=\begin{cases}1, & 0\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$$

进一步，如果 $0\leqslant x\leqslant 1$，则有：

$$F(x)=\int_0^x f(y)\mathrm{d}y=\int_0^x 1\mathrm{d}y=x$$ ◀

[如果 $x<0$ 或者 $x>1$，$F(x)$是多少呢?]$f(x)$和 $F(x)$的图形分别如图 4.4 和图 4.5 所示。

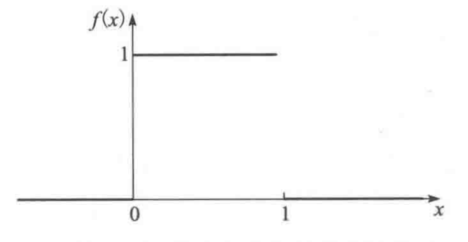


图 4.4　[0, 1]区间上的均匀随机变量的 $f(x)$

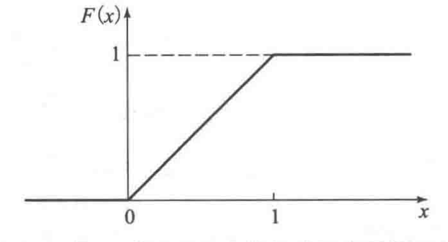


图 4.5　[0, 1]区间上的均匀随机变量的 $F(x)$

最后，如果 $0\leqslant x<x+\Delta x\leqslant 1$，则有：

$$P(X\in[x,x+\Delta x])=\int_x^{x+\Delta x} f(y)\mathrm{d}y=F(x+\Delta x)-F(x)=(x+\Delta x)-x=\Delta x$$

因此，[0, 1]上的均匀随机变量以相同的可能性落在 0 和 1 之间长度为 Δx 的任意区间内，这也证明了其名字“均匀”是有道理的。[0, 1]上的均匀分布的随机变量对仿真来说是基础性的，因为它是在计算机中产生任何随机变量的基础(见第 7、8 章)。

例 4.8 在第 1 章中，指数随机变量在排队的例子中用于描述到达间隔时间和服务时间，在库存例子中用于描述需求间隔时间。均值为 β 的指数随机变量的概率密度函数和分布函数如图 4.6 和图 4.7 所示。

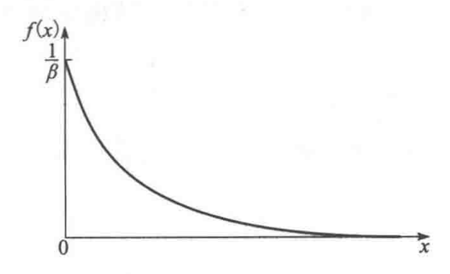


图 4.6 均值为 β 的指数随机变量的 $f(x)$

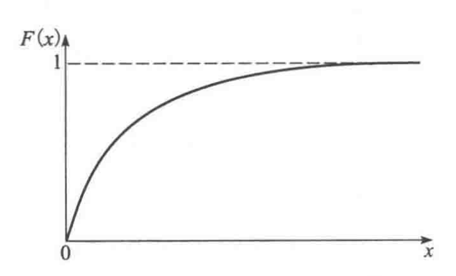


图 4.7 均值为 β 的指数随机变量的 $F(x)$ ◀

本章到目前为止我们一次只考虑一个随机变量，但是在仿真中经常必须同时处理 n(一个正整数)个随机变量 X_1，X_2，…，X_n。例如，在第 1.4 节的排队模型中，我们关心(输入)服务时间随机变量 S_1，S_2，…，S_n 和(输出)延误时间随机变量 D_1，D_2，…，D_n。在随后的讨论中，为了说明方便，我们假设 $n=2$，并且问题中的两个随机变量是 X 和 Y。

如果 X 和 Y 是离散随机变量，则令

$$p(x,y)=P(X=x,Y=y),\quad \text{对于 } x,y \text{ 成立}$$

其中，$p(x, y)$称为 X 和 Y 的联合概率质量函数。

在这种情况下，X 和 Y 是独立的，如果满足

$$p(x,y)=p_X(x)p_Y(y),\quad \text{对于所有 } x,y \text{ 成立}$$

其中，

$$P_X(x)=\sum_{\text{所有}y}p(x,y)$$

$$P_Y(y)=\sum_{\text{所有}x}p(x,y)$$

是 X 和 Y 的(边际)概率质量函数。

例 4.9 假设 X 和 Y 是联合离散随机变量，满足

$$p(x,y)=\begin{cases}\dfrac{xy}{27}, & x=1,2 \text{ 且 } y=2,3,4\\ 0, & \text{其他}\end{cases}$$

则有：

$$p_X(x)=\sum_{y=2}^{4}\frac{xy}{27}=\frac{x}{3},\quad x=1,2$$

$$p_Y(x)=\sum_{x=1}^{2}\frac{xy}{27}=\frac{y}{9},\quad y=2,3,4$$

由于对于所有的 x，y，满足 $p(x, y)=xy/27=p_X(x)p_Y(y)$，因此随机变量 X 和 Y 是独立的。 ◀

例 4.10 设无放回地从 52 张纸牌中抽取 2 张牌。令随机变量 X 和 Y 分别表示出现 A 和国王的次数，则两者的可能取值为 0，1，2。可以由下式表示：

$$p_X(1)=p_Y(1)=2\times\left(\frac{4}{52}\right)\times\left(\frac{48}{51}\right)$$

且

$$p(1,1)=2\times\left(\frac{4}{52}\right)\times\left(\frac{4}{51}\right)$$

由于

$$p(1,1) = 2\times\left(\frac{4}{52}\right)\times\left(\frac{4}{51}\right)\neq 4\times\left(\frac{4}{52}\right)^2\times\left(\frac{48}{51}\right)^2$$

因此，X 和 Y 是非独立的(见习题 4.5)。 ◀

如果存在一个非负函数 $f(x, y)$(称为 X 和 Y 的联合概率密度函数)，称随机变量 X 和 Y 是联合连续的，对于所有的实数集合 A 和 B，满足

$$P(X\in A, Y\in B) = \int_B\int_A f(x,y)\mathrm{d}x\mathrm{d}y$$

在这种情况下，如果满足

$$f(x,y) = f_X(x)f_Y(y),\quad \text{对于所有 } x,y$$

则 X 和 Y 是独立的，其中，

$$f_X(x) = \int_{-\infty}^{+\infty} f(x,y)\mathrm{d}y$$

$$f_Y(y) = \int_{-\infty}^{+\infty} f(x,y)\mathrm{d}x$$

分别是 X 和 Y 的(边际)概率密度函数。

例 4.11 假设 X 和 Y 是联合连续随机变量，满足

$$f(x,y) = \begin{cases} 24xy, & \text{对于 } x\geqslant 0, y\geqslant 0, \text{且 } x+y\leqslant 1 \\ 0, & \text{其他} \end{cases}$$

则

$$f_X(x) = \int_0^{1-x} 24xy\mathrm{d}y = 12xy^2\big|_0^{1-x} = 12x(1-x)^2,\quad 0\leqslant x\leqslant 1$$

$$f_Y(x) = \int_0^{1-y} 24xy\mathrm{d}y = 12yx^2\big|_0^{1-y} = 12y(1-y)^2,\quad 0\leqslant y\leqslant 1$$

由于

$$f\left(\frac{1}{2},\frac{1}{2}\right) = 6 \neq \left(\frac{3}{2}\right)^2 = f_X\left(\frac{1}{2}\right)f_Y\left(\frac{1}{2}\right)$$

因此 X 和 Y 是非独立的。 ◀

由此，如果知道了一个随机变量的取值无法得到另一个的分布，则随机变量 X 和 Y(无论是离散或是连续)是独立的。而且，如果 X 和 Y 是非独立的，则称它们为相关的。

现在再一次考虑有 n 个随机变量 X_1，X_2，…，X_n 的情形，讨论单一随机变量 X_i 的特征以及两个随机变量 X_i 和 X_j 之间可能存在相关性的某些度量。

随机变量 $X_i(i=1, 2, \cdots, n)$ 的均值或者期望值记为 μ_i 或者 $E(X_i)$，定义为：

$$\mu_i = \begin{cases} \sum_{j=1}^{+\infty} x_j p_{X_i}(x_j), & \text{对于 } X_i \text{ 是离散的} \\ \int_{-\infty}^{+\infty} xf_{X_i}(x)\mathrm{d}x, & \text{对于 } X_i \text{ 是连续的} \end{cases}$$

均值是在重心意义上的集中趋势的度量[见，例如，Billingsley 等(1986，第 42-43 页)]。

例 4.12 在例 4.5 中需求量随机变量的均值为：

$$\mu = 1\times\left(\frac{1}{6}\right)+2\times\left(\frac{1}{3}\right)+3\times\left(\frac{1}{3}\right)+4\times\left(\frac{1}{6}\right) = \frac{5}{2}$$ ◀

例 4.13 例 4.7 中均匀随机变量的均值为：

$$\mu = \int_0^1 xf(x)\mathrm{d}x = \int_0^1 x\mathrm{d}x = \frac{1}{2}$$

用 c 或者 c_i 表示一个常数(实数)，则下面是均值的一些重要性质。

(1) $E(cX) = cE(X)$。

(2) $E\left(\sum_{i=1}^{n} c_i X_i\right) = \sum_{i=1}^{n} c_i E(X_i)$，即使 X_i 之间是相关的。◀

随机变量 X_i 的中值 $x_{0.5}$ 是集中趋势的另一个度量，定义为满足 $F_{X_i}(x) \geqslant 0.5$ 的 x 的最小值。如果 X_i 是一个连续随机变量，则 $F_{X_i}(x_{0.5}) = 0.5$，如图 4.8 所示。当 X_i 取值可能非常大或者非常小时，中值可能是一个比均值更好地对集中趋势的度量，因为如果极值发生的可能性即使非常小，它也会极大地影响均值，而中值不会。

例 4.14 考虑离散随机变量 X，它以概率 0.2 取 1，2，3，4，5 每个值。很明显，X 的均值和中值都是 3。现在考虑随机变量 Y，它取值为 1，2，3，4，以及 100 的每个的概率是 0.2。Y 的均值和中值分别是 22 和 3，注意中值对于分布的这种变化是不敏感的。◀

连续(离散)随机变量 X_i 的众数 m 是关于集中趋势的另一个度量，定义为使 $f_{X_i}(x)[p_{X_i}(x)]$ 最大的 x 值(见图 4.8)。注意，众数对于某些分布可能不是唯一的。

随机变量 X_i 的方差用 σ_i^2 或者 $\text{var}(X_i)$ 表示，由下式定义：

$$\sigma_i^2 = E[(X_i - \mu_i)^2] = E(X_i^2) - \mu_i^2$$

方差是随机变量关于其均值离差的度量，如图 4.9 所示。方差越大，随机变量取值越可能远离其均值。

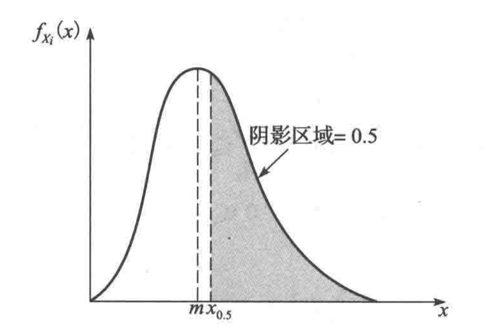


图 4.8 连续随机变量的中值 $x_{0.5}$ 和众数 m

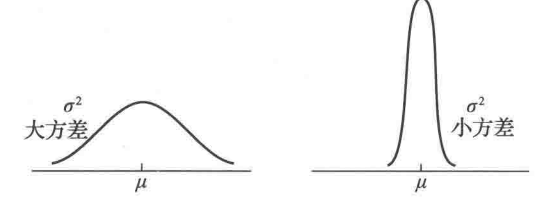


图 4.9 具有大方差和小方差的连续随机变量的密度函数

例 4.15 对例 4.5 中的需求量随机变量，其方差计算如下：

$$E(X^2) = 1^2 \times \left(\frac{1}{6}\right) + 2^2 \times \left(\frac{1}{3}\right) + 3^2 \times \left(\frac{1}{3}\right) + 4^2 \times \left(\frac{1}{6}\right) = \frac{43}{6}$$

$$\text{var}(X) = E(X^2) - \mu^2 = \frac{43}{6} - \left(\frac{5}{2}\right)^2 = \frac{11}{12}$$ ◀

例 4.16 对例 4.7 中的[0，1]区间上的均匀随机变量，其方差计算如下：

$$E(X^2) = \int_0^1 x^2 f(x)\,\mathrm{d}x = \int_0^1 x^2\,\mathrm{d}x = \frac{1}{3}$$

$$\text{var}(X) = E(X^2) - \mu^2 = \frac{1}{3} - \left(\frac{1}{2}\right)^2 = \frac{1}{12}$$ ◀

方差具有如下性质：

(1) $\text{var}(X) \geqslant 0$。

(2) $\text{var}(cX) = c^2 \text{var}(X)$。

(3) 如果 X_i 是相互独立的(或者是不相关的，如下述讨论)，则 $\text{var}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} \text{var}(X_i)$。

随机变量 X_i 的标准差定义为 $\sigma_i = \sqrt{\sigma_i^2}$。当 X_i 满足正态分布时，标准差能给出最明确

的解释(见第 6.2.2 小节)。特别地，假设 X_i 满足均值为 μ_i、标准差为 σ_i 的正态分布。在这种情况下，举例来说，X_i 在 $\mu_i-1.96\sigma_i$ 和 $\mu_i+1.96\sigma_i$ 之间的概率为 0.95。

现在考虑两个随机变量之间相关性的度量。两个随机变量 X_i 和 X_j(其中 $i=1$, 2, …, n; $j=1$, 2, …, n)之间的协方差是它们(线性)相关性的度量，用 C_{ij} 或者 $\text{cov}(X_i, Y_j)$表示，定义为

$$C_{ij} = E[(X_i-\mu_i)(X_j-\mu_j)] = E(X_iX_j) - \mu_i\mu_j \tag{4.1}$$

注意，协方差是对称的，即 $C_{ij}=C_{ji}$，而且，当 $i=j$ 时，$C_{ij}=C_{ii}=\sigma_i^2$。

例 4.17 对例 4.11 中的联合连续随机变量 X 和 Y，其协方差计算如下：

$$E(XY) = \int_0^1\int_0^{1-x} xyf(x,y)\,\mathrm{d}y\mathrm{d}x = \int_0^1 x^2\left(\int_0^{1-x}24y^2\,\mathrm{d}y\right)\mathrm{d}x$$

$$= \int_0^1 8x^2(1-x)^3\,\mathrm{d}x = \frac{2}{15}$$

$$E(X) = \int_0^1 xf_X(x)\,\mathrm{d}x = \int_0^1 12x^2\ (1-x)^2\,\mathrm{d}x = \frac{2}{5}$$

$$E(Y) = \int_0^1 yf_Y(y)\,\mathrm{d}y = \int_0^1 12y^2\ (1-y)^2\,\mathrm{d}y = \frac{2}{5}$$

因此

$$\text{cov}(X,Y) = E(XY) - E(X)E(Y) = \frac{2}{15} - \left(\frac{2}{5}\right)\times\left(\frac{2}{5}\right) = -\frac{2}{75}$$

如果 $C_{ij}=0$，则称随机变量 X_i 和 X_j 是不相关的。易于证明，如果 X_i 和 X_j 是独立的随机变量，则 $C_{ij}=0$(见习题 4.8)。可是，一般来说，其逆命题是非真的(见习题 4.9)。然而，如果 X_i 和 X_j 是具有 $C_{ij}=0$ 的联合正态分布的随机变量，则它们也是独立的(见习题 4.10)。 ◀

现在给出两个定义来阐述协方差 C_{ij} 的含义。如果 $C_{ij}>0$，则称 X_i 和 X_j 是正相关的。在这种情况下，$X_i>\mu_i$ 和 $X_j>\mu_j$ 倾向于同时发生，$X_i<\mu_i$ 和 $X_j<\mu_j$ 也倾向于同时发生[见式(4.1)]。因此，对正相关的随机变量来说，如果一个大，另一个很可能也大。如果 $C_{ij}<0$，则称 X_i 和 X_j 是负相关的。在这种情况下，$X_i>\mu_i$ 和 $X_j<\mu_j$ 倾向于同时发生，$X_i<\mu_i$ 和 $X_j>\mu_j$ 也倾向于同时发生。因此，对负相关的随机变量来说，如果一个大，另一个很可能小。我们在下一节中给出正相关和负相关随机变量的例子。

如果 X_1, X_2, …, X_n 是仿真输出数据(例如，X_i 可能是第 1.4 节中排队例子的延误时间 D_i)，对于 $i=1$, 2, …, n，我们经常不仅仅需要知道均值 μ_i 和方差 ${\sigma_i}^2$，还需要知道 X_i 和 $X_j(i\neq j)$之间相关性的度量。然而，把 C_{ij} 作为 X_i 和 X_j 之间相关性的度量的困难性在于它并不是无量纲的，这使得它存在解释上的困难(譬如说，如果 X_i 和 X_j 的单位是分钟，则 C_{ij} 的单位是分钟的平方)。因此，我们引入相关系数 ρ_{ij}，定义为

$$\rho_{ij} = \frac{C_{ij}}{\sqrt{\sigma_i^2\sigma_j^2}},\quad \begin{matrix} i=1,2,\cdots,n \\ j=1,2,\cdots,n \end{matrix} \tag{4.2}$$

作为 X_i 和 X_j 之间(线性)相关性(见习题 4.11)的主要度量[将 X_i 和 X_j 之间相关系数也记为 $\text{cor}(X_i, X_j)$]。由于式(4.2)的分母是正的，显然 ρ_{ij} 与 C_{ij} 符号相同。此外，可以证明对于所有的 i 和 j，满足 $-1\leqslant\rho_{ij}\leqslant 1$(见习题 4.12)。如果 ρ_{ij} 接近 $+1$，则 X_i 和 X_j 是高度正相关的。反之，如果 ρ_{ij} 接近 -1，则 X_i 和 X_j 是高度负相关的。

例 4.18 对于例 4.11 中的随机变量，可以证明 $\text{var}(X)=\text{var}(Y)=\frac{1}{25}$。因此

$$\text{cor}(X,Y) = \frac{\text{cov}(X,Y)}{\sqrt{\text{var}(X)\text{var}(Y)}} = \frac{-\frac{2}{75}}{\frac{1}{25}} = -\frac{2}{3}$$ ◀

4.3 仿真输出数据和随机过程

由于大部分的仿真模型采用随机变量作为输入，仿真输出数据自身也是随机的，因此在给出有关模型的真实特征的结论时需要非常谨慎，例如，第 1.4 节中排队例子的(期望)平均延误时间。在本节和接下来的三节我们为第 9 章和第 10 章中输出数据分析的谨慎处理打下基础。

随机过程是按时间排序的“类似的”随机变量的集合，随机变量全部定义在共同样本空间上。这些随机变量可以取的所有可能值的集合称为状态空间。如果这个集合是 X_1，X_2，…，则我们有一个离散时间随机过程。如果这个集合是$\{X(t), t\geqslant 0\}$，则我们有一个连续时间随机过程。

例 4.19 考虑一个单服务台排队系统，例如，$M/M/1$ 队列，具有独立同分布的到达间隔时间 A_1，A_2，…，独立同分布的服务时间 S_1，S_2，…，顾客按照先进先出方式服务。对于生成随机变量 A_1，A_2，…和 S_1，S_2，…的实验，人们可以定义在队列中的延误时间 D_1，D_2，…的离散随机过程为(见习题 4.14)：

$$D_1 = 0$$

$$D_{i+1} = \max\{D_i + S_i - A_{i+1}, 0\}, \quad i = 1,2,\cdots$$

因此，仿真将输入随机变量(例如，A_i 和 S_i)映射到有关的输出随机过程 D_1，D_2，…。这里，状态空间是非负实数集。注意，D_i 和 D_{i+1} 是正相关的(为什么)。◀

例 4.20 对于例 4.19 的排队系统，令 $Q(t)$ 为在 t 时刻在队列中的顾客数，则$\{Q(t), t\geqslant 0\}$是一个连续时间随机过程，其状态空间为$\{0, 1, 2, \cdots\}$。◀

例 4.21 对于第 1.5 节的库存系统，令 C_i 为在第 i 个月的总成本(例如，订货、存储和缺货成本的总和)，则 C_1，C_2，…是离散时间随机过程，其状态空间是非负实数。◀

为了由一组仿真输出数据推断出一个隐含的随机过程，人们有时必须做一些关于随机过程的假设，这些假设在实际中也许并不严格地成立(然而，如果没有这些假设就可能无法进行输出数据的统计分析)。一个这样的例子是假设一个随机过程是协方差平稳的，该性质是我们现在定义的。一个离散时间随机过程 X_1，X_2，…称为是协方差平稳的，如果

$$\mu_i = \mu, \quad i = 1,2,\cdots \text{ 且 } -\infty < \mu < +\infty$$

$$\sigma_i^2 = \sigma^2, \quad i = 1,2,\cdots \text{ 且 } \sigma^2 < +\infty$$

并且对于 $j=1, 2, \cdots$，$C_{i,i+j}=\text{cov}(X_i, X_{i+j})$是关于 i 独立的。

因此，对于一个协方差平稳过程，均值和方差是不随时间变化而变化的(这个共同的均值和方差分别标记为 μ 和 σ^2)，且两个观测值 X_i 和 X_{i+j}之间的协方差仅仅取决于间隔 j(有时称为迟后)，而不取决于实际时间值 i 和 $i+j$(也可以以类似的方法定义一个协方差平稳连续时间随机过程$\{X(t), t\geqslant 0\}$)。

对于一个协方差平稳过程，分别将 X_i 和 X_{i+j}之间的协方差和相关系数用 C_j 和 ρ_j 表示，其中，

$$\rho_j = \frac{C_{i,i+j}}{\sqrt{\sigma_i^2 \sigma_{i+j}^2}} = \frac{C_j}{\sigma^2} = \frac{C_j}{C_0}, \quad j = 0,1,2,\cdots$$

例 4.22 考虑输出过程为 D_1，D_2，…是协方差平稳的(该技术细节讨论见附录 4A)$M/M/1$ 队列，满足 $\rho=\lambda/\omega<1$(回忆一下，λ 是到达速率，ω 是服务速率)。从 Daley (1968)的结果可以计算出 ρ_j，我们在图 4.10 中绘出 $\rho=0.5$ 和 $\rho=0.9$ 的情况(不要混淆 ρ_j 和 ρ)。注意相关系数 ρ_j 是正的并随着 j 的增加单调递减到 0。特别地，当 $\rho=0.9$ 时，$\rho_1=0.99$，当 $\rho=0.5$ 时，$\rho_1=0.78$。此外，$\rho=0.9$ 时，ρ_j 收敛到 0 的速度明显较慢；事实上，令人惊讶的，ρ_{50} 是 0.69(一般说来，我们的实验表明，排队系统的输出过程是正相关的)。

◀

例 4.23 考虑一个0交货延迟、0积压的(s, S)库存系统(相较于第1.5节所给出的例子，该库存系统是其简化版，采用解析方法计算所需的相关性)。令I_i，J_i和Q_i分别为在第i月时，未被订购之前的库存量、订购之后的库存量，以及需求量。假设Q_i服从均值为25的泊松分布(详细讨论参见第6.2.3小节)，即：

$$p(x) = P(Q_i = x) = \frac{e^{-25}(25)^x}{x!}, x = 0,1,2,\cdots$$

如果$I_i < s$，我们订购$S - I_i$件商品$(J_i = S)$，产生的费用为$K + i(S - I_i)$，其中，$K = 32$，$i = 3$。如果$I_i \geqslant s$，则不进行订购$(J_i = I_i)$，亦不产生订购费用。J_i确定后，产生需求量Q_i。如果$J_i - Q_i \geqslant 0$，则产生库存费用$h(J_i - Q_i)$，其中$h = 1$。如果$J_i - Q_i < 0$，则产生缺货费用$\pi(Q_i - J_i)$，其中$\pi = 5$。无论哪种情况下，均有$I_{i+1} = J_i - Q_i$。令C_i为第i个月的总费用，假设$s = 17$，$S = 57$，$I_1 = S$。根据Wagner(1969，第A19页)的结果，可以计算出输出过程C_1，C_2，…的ρ_j，如图4.11所示(技术细节的讨论见附录4A)。注意ρ_2的值为正，因为对于这个特定系统来说，每月订购下个月的货，导致每次开销很大。另一方面，ρ_1的值为负，因为如果在指定的月订购(大开销)，则下个月很可能不订购(小开销)。

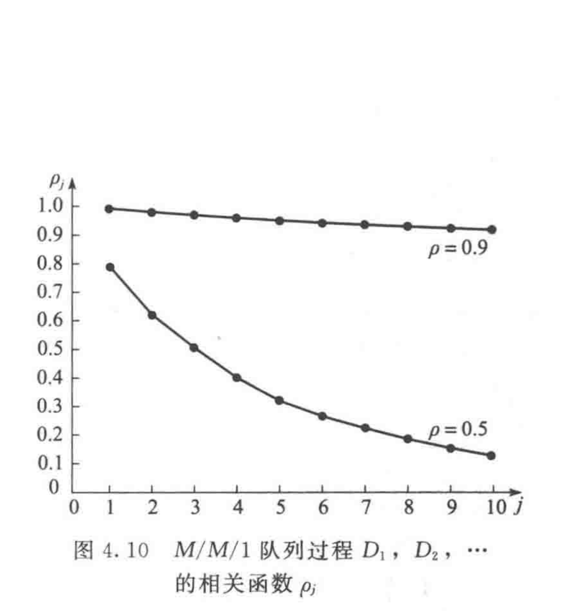


图4.10 $M/M/1$队列过程D_1，D_2，…的相关函数ρ_j

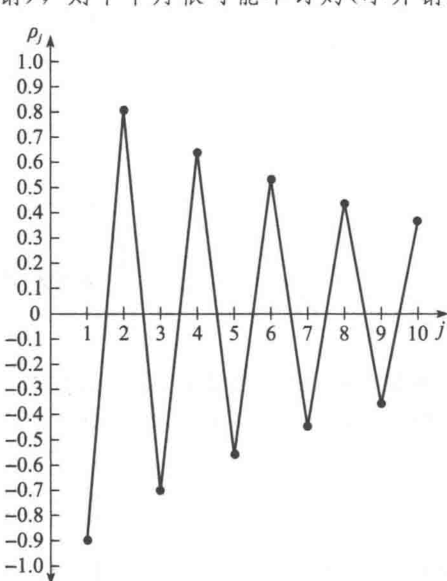


图4.11 (s, S)库存系统过程C_1，C_2，…的相关函数ρ_j ◀

如果X_1，X_2，…是仿真中在0时刻开始的统计过程，则其很可能不是协方差平稳的(见附录4A)。然而，在某些仿真中，如果k足够大，X_{k+1}，X_{k+2}，…是接近协方差平稳的，其中k是预热周期的长度(见第9.5.1小节)。

4.4 均值、方差和相关系数的估计

假设X_1，X_2，…，X_n是独立同分布的随机变量(观测值)，有限总体均值为μ，有限总体方差为σ^2，我们的主要目标是估计μ；对σ^2的估计是第二个关心的。则样本均值为：

$$\overline{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n} \tag{4.3}$$

这是μ的一个无偏(点)估计；也就是说，$E[\overline{X}(n)] = \mu$(见习题4.16)[直观地说，$\overline{X}(n)$是μ的一个无偏估计，意味着如果做大量独立实验，每次都得到一个$\overline{X}(n)$，这一系列$\overline{X}(n)$的平均值就是μ]。类似地，样本方差为：

$$S^2(n)=\frac{\sum_{i=1}^{n}[X_i-\overline{X}(n)]^2}{n-1} \tag{4.4}$$

这是 σ^2 的一个无偏估计，因为 $E[S^2(n)]=\sigma^2$（见习题 4.16）。注意，估计 $\overline{X}(n)$ 和 $S^2(n)$ 有时分别用 $\hat{\mu}$ 和 $\hat{\sigma}^2$ 表示。

如果没有任何附加信息，使用 $\overline{X}(n)$ 作为 μ 的估计比较困难，原因在于我们没有办法来评定 $\overline{X}(n)$ 有多接近于 μ。因为 $\overline{X}(n)$ 是一个方差为 $\text{var}[\overline{X}(n)]$ 的随机变量，在一次实验中，$\overline{X}(n)$ 可能很接近 μ，在另一次实验中，$\overline{X}(n)$ 可能与 μ 相差很远（见图 4.12，其中一系列 X_i 假定为连续随机变量）。评定 $\overline{X}(n)$ 作为 μ 的估计的精确度的一个常用的方法是，建立一个 μ 的置信区间，这部分内容我们在下一节中讨论。然而，建立一个置信区间的第一步是估计 $\text{var}[\overline{X}(n)]$。因为

$$\begin{aligned}\text{var}[\overline{X}(n)]&=\text{var}\left(\frac{1}{n}\sum_{i=1}^{n}X_i\right)=\frac{1}{n^2}\text{var}\left(\sum_{i=1}^{n}X_i\right)\\&=\frac{1}{n^2}\sum_{i=1}^{n}\text{var}(X_i)=\frac{1}{n^2}n\sigma^2=\frac{\sigma^2}{n}\quad（因为 X_i 之间是独立的）\end{aligned} \tag{4.5}$$

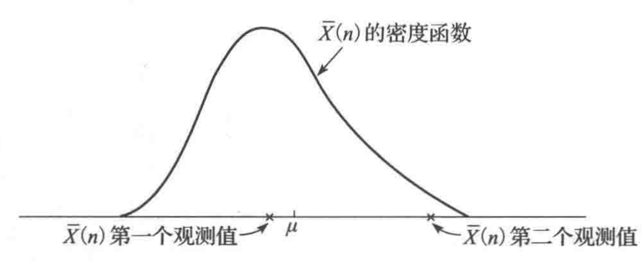


图 4.12 随机变量 $\overline{X}(n)$ 的两个观测值

显然，一般情况下，样本长度 n 越大，$\overline{X}(n)$ 应该越接近 μ（见图 4.13）。此外，$\text{var}[\overline{X}(n)]$ 的无偏估计可以通过用 $S^2(n)$ 替换式(4.5)的 σ^2 获得，结果为：

$$\widehat{\text{var}}[\overline{X}(n)]=\frac{S^2(n)}{n}=\frac{\sum_{i=1}^{n}[X_i-\overline{X}(n)]^2}{n(n-1)}$$

观察 $\widehat{\text{var}}[\overline{X}(n)]$ 的表达式，当它用 X_i 和 $\overline{X}(n)$ 的形式重写时，分母中包含 n 和 $n-1$。

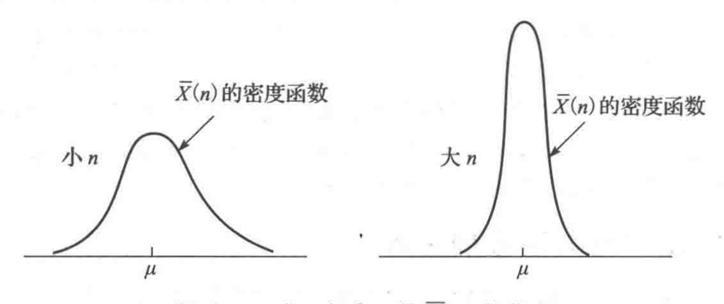


图 4.13 小 n 与大 n 的 $\overline{X}(n)$ 的分布

最后，注意，如果 X_i 是独立的，它们是不相关的，且对于 $j=1, 2, \cdots, n-1$，有 $\rho_j=0$。

我们的经验是仿真输出数据几乎总是相关的。因此，上述关于独立同分布观测值的讨论不能直接应用于分析仿真输出数据。为了理解将仿真输出数据按它们独立来处理的危险，我们使用在上一节中讨论的协方差平稳模型。特别地，假设随机变量 X_1，X_2，…，X_n 来自一个协方差平稳的随机过程。那么，样本均值 $\overline{X}(n)$ 是 μ 的一个无偏估计值仍然是

正确的；然而，样本方差 $S^2(n)$ 不再是 σ^2 的无偏估计值了。事实上，可以证明[参见 Anderson(1994，第 448 页)]

$$E[S^2(n)] = \sigma^2 \left[1 - 2\frac{\sum_{j=1}^{n-1}(1-j/n)\rho_j}{n-1}\right] \tag{4.6}$$

因此，如果 $\rho_j > 0$（正相关），作为实际中非常常见的情况，$S^2(n)$ 会有一个负偏差：$E[S^2(n)] < \sigma^2$。这是非常重要的，因此不要以为有一些仿真软件产品采用 $E[S^2(n)]$ 去评估一组仿真输出数据的方差，这在分析中会导致严重的错误。

现在考虑当 X_1，X_2，…，X_n 来自一个协方差平稳过程时，估计样本均值的方差 $\text{var}[\overline{X}(n)]$ 的问题(这个值在下一节用于构建 μ 的置信空间)。可以证明(见习题 4.17)

$$\text{var}[\overline{X}(n)] = \sigma^2 \frac{\left[1 + 2\sum_{j=1}^{n-1}(1-j/n)\rho_j\right]}{n} \tag{4.7}$$

因此，如果我们通过 $S^2(n)/n$ 来估计 $\text{var}[\overline{X}(n)]$(在独立同分布的情况下是正确的表达式)，过去常常这样做，存在两个误差来源：$S^2(n)$ 作为 σ^2 的估计的偏差以及忽略了式(4.7)中相关系数项。事实上，如果合并等式(4.6)和等式(4.7)，可得到：

$$E\left[\frac{S^2(n)}{n}\right] = \frac{[n/a(n)]-1}{n-1}\text{var}[\overline{X}(n)] \tag{4.8}$$

其中，$a(n)$ 表示式(4.7)中方括号中的量。

若 $\rho_j > 0$ 则 $a(n) > 1$，且 $E[S^2(n)/n] < \text{var}[\overline{X}(n)]$。

例 4.24 假设有 $\rho=0.9$ 的协方差平稳 $M/M/1$ 队列的延误过程 D_1，D_2，…中的数据 D_1，D_2，…，D_{10}。那么，将实际相关系数 ρ_j（其中 $j=1$，2，…，9）代入式(4.6)和式(4.8)，得到

$$E[S^2(10)] = 0.0328\sigma^2$$

且

$$E\left[\frac{S^2(10)}{10}\right] = 0.0034\text{var}[\overline{D}(10)]$$

其中，

$$\sigma^2 = \text{var}(D_i);\quad \overline{D}(10) = \frac{\sum_{i=1}^{10} D_i}{10};\quad S^2(10) = \frac{\sum_{i=1}^{10}[D_i - \overline{D}(10)]^2}{9}$$

因此，平均说来，$\frac{S^2(10)}{10}$ 是对 $\text{var}[\overline{D}(10)]$ 的严重低估，我们关于 $\overline{D}(10)$ 很接近 $\mu = E(D_i)$ 的看法过于乐观了。◀

有时人们对从数据 X_1，X_2，…，X_n 来估计 ρ_j（或者 C_j）感兴趣{例如，ρ_j 的估计值可能会被代入等式(4.7)来获得 $\text{var}[\overline{X}(n)]$ 的一个较好估计值；应用见第 9.5.3 小节}。在这种情况下，$\rho_j(j=1, 2, \cdots, n-1)$ 可由下式估计：

$$\hat{\rho}_j = \frac{\hat{C}_j}{S^2(n)},\quad \hat{C}_j = \frac{\sum_{i=1}^{n-j}[X_i - \overline{X}(n)][X_{i+j} - \overline{X}(n)]}{n-j} \tag{4.9}$$

ρ_j 也用其他估计值。例如，人们可以用 n 替换 $\hat{C}_j$ 式分母中的 $n-j$。采用估计值 $\hat{\rho}_j$（或者 ρ_j 的任何其他估计值)的困难在于它是有偏的，它会有一个很大的方差，除非 n 非常大，而且它是与其他相关系数估计值相关的，即 $\text{cov}(\hat{\rho}_j, \hat{\rho}_k) \neq 0${特别地，$\hat{\rho}_{n-1}$ 会是 ρ_{n-1} 的一个较差的估计值，因为它是基于一次乘 $[X_1 - \overline{X}(n)][X_n - \overline{X}(n)]$}。因此，一般来说，$\rho_j$ 的"好的"估计值是很难获得的，除非 n 非常大且 j 相对于 n 很小。

例 4.25 假设我们有例 4.24 中的过程数据 D_1，D_2，…，D_{100}。在图 4.14 中，对于 $j=1$，2，…，10，绘出$\hat{\rho}_j$[由式(4.9)计算]和 ρ_j。注意相关系数的估计的质量较差。◀

注意，当 X_i 独立时，相关系数的估计不必要一定是 0，因为估计值$\hat{\rho}_j$ 是一个随机变量。

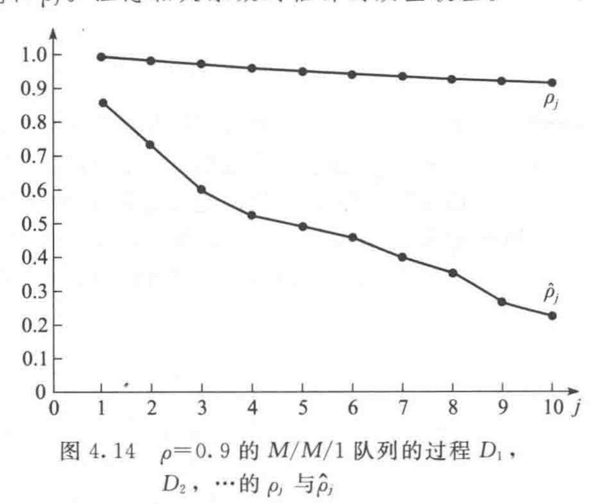


图 4.14 $\rho=0.9$ 的 $M/M/1$ 队列的过程 D_1，D_2，…的 ρ_j 与$\hat{\rho}_j$

我们已经看到仿真输出数据是相关的，因此基于独立同分布观测值的经典统计学公式不能直接用于估计方差。然而，在第 9 章中会看到往往能将仿真输出数据分组成新的"观测值"，对这些新的"观察值"可以应用基于独立同分布观测值的公式。因此，这节和下两节中的基于独立同分布观测值的公式可以间接地应用于分析仿真输出数据。

4.5 均值的置信区间和假设检验

令 X_1，X_2，…，X_n 为独立同分布随机变量，具有有限均值 μ 和有限方差 σ^2(还假设 $\sigma^2>0$，这样 X_i 不是退化的随机变量)。在这节中将讨论如何构建 μ 的置信区间以及检验假设 $\mu=\mu_0$ 互补问题。

首先说明概率论中最重要的结果，经典中心极限定理。令 Z_n 为随机变量$[\overline{X}(n)-\mu]/\sqrt{\sigma^2/n}$，且令 $F_n(z)$是样本大小为 n 的 Z_n 的分布函数；即 $F_n(z)=P(Z_n\leqslant z)$[注意，μ 和 σ^2/n 分别是 $\overline{X}(n)$的均值和方差]。则中心极限定理如下[证明见 Chung(1974，第 169 页)]。

定理 4.1 当 $n\to+\infty$时 $F_n(z)\to\Phi(z)$，其中 $\Phi(z)$是 $\mu=0$ 且 $\sigma^2=1$ 的正态随机变量的分布函数(此后称为标准正态随机变量，见第 6.2.2 小节)，由下式给出：

$$\Phi(z)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z}\mathrm{e}^{-y^2/2}\mathrm{d}y,\ -\infty<z<+\infty$$

该定理说明，如果 n"足够大"，随机变量 Z_n 的分布将近似标准正态随机变量分布，不管 X_i 实际是何种分布。还可以证明，对于一个很大的 n，样本均值 $\overline{X}(n)$的分布近似均值为 μ 和方差为 σ^2/n 的正态随机变量。

在实际中应用上述结果的困难在于方差 σ^2 一般是不知道的。虽然如此，由于样本方差 $S^2(n)$随着 n 的增大而收敛于σ^2，可以证明，如果用 $S^2(n)$代替 Z_n 表达式中的σ^2，定理 4.1 仍然是成立的。利用这个改变，如果 n 足够大，随机变量 $t_n=[\overline{X}(n)-\mu]/\sqrt{S^2(n)/n}$ 的分布近似标准正态随机变量。对于大的 n，就有：

$$\begin{aligned}&P\left(-z_{1-\alpha/2}\leqslant\frac{\overline{X}(n)-\mu}{\sqrt{S^2(n)/n}}\leqslant z_{1-\alpha/2}\right)\\&=P\left[\overline{X}(n)-z_{1-\alpha/2}\sqrt{\frac{S^2(n)}{n}}\leqslant\mu\leqslant\overline{X}(n)+z_{1-\alpha/2}\sqrt{\frac{S^2(n)}{n}}\right]\\&\approx1-\alpha\end{aligned}\tag{4.10}$$

其中，符号≈的含义是"约等于"；$z_{1-\alpha/2}(0<\alpha<1)$是标准正态随机变量上 $1-\alpha/2$ 临界点(见图 4.15 和本书末尾附录的表 A.1 最后一行)。因此，如果 n 足够大，μ 的近似 $100(1-\alpha)\%$ 的置信区间为：

$$\overline{X}(n)\pm z_{1-\alpha/2}\sqrt{\frac{S^2(n)}{n}}\tag{4.11}$$

对于给定的一组数据 X_1，X_2，…，X_n，置信区间的下界 $l(n, \alpha)=\overline{X}(n)-z_{1-\alpha/2}\sqrt{\frac{S^2(n)}{n}}$ 和置信区间上界 $u(n, \alpha)=\overline{X}(n)+z_{1-\alpha/2}\sqrt{\frac{S^2(n)}{n}}$ 都是数值(实际上，是随机变量的特定实现)，而置信区间$[l(n, \alpha), u(n, \alpha)]$要么包含 μ，要么不包含 μ。因此，在获取到数据且给出区间的端点值之后，单一置信区间$[l(n, \alpha), u(n, \alpha)]$就没有什么随机性了。对于置信区间式(4.11)的正确解释如下[见式(4.10)]：如果构建了很大数目的独立的 $100(1-\alpha)\%$置信区间，每个都基于 n 个观察值，其中 n 足够大，这些置信区间包含(覆盖)μ 的比例应该是 $1-\alpha$。我们称这个比例为置信区间的覆盖度。

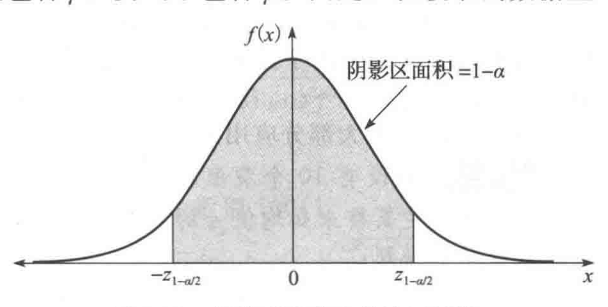


图 4.15　标准正态分布的密度函数

例 4.26　为了进一步详述置信区间的正确解释，我们从均值为 5，方差为 1 的一个正态分布中生成了 15 个独立样本，样本长度为 $n=10$。对于任意一个数据集，构建一个 μ 的 90% 的置信区间，其中 μ 的值已知为 5。在图 4.16 中垂直绘制了 15 个置信区间(位于置信区间中心的点为样本均值)，图中可见，除第 7 个和第 13 个样本点外，所有的置信区间所覆盖的均值高度均为 5。通常，如果我们想要构建大量的置信度为 90% 的置信区间，则会发现其中 90% 的置信区间一定会包含(覆盖)μ。◀

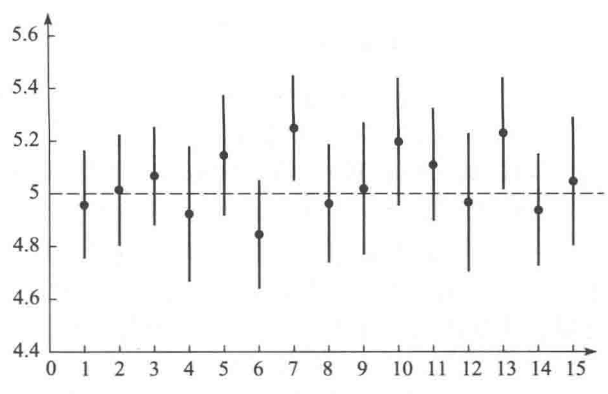


图 4.16　基于 $n=10$ 个观察值生成的置信区间，这些观测值来自于一个均值为 5，方差为 1 的正态分布

应用式(4.11)构建 μ 的置信区间的困难在于知道“n 足够大”的含义。结果发现，X_i 的实际分布越偏(非对称)，t_n 的分布近似于 $\Phi(z)$所需要的 n 值就越大(见本节中后面的讨论)。如果 n 选得太小，对所要求的 $100(1-\alpha)\%$置信区间的实际覆盖度一般会小于 $1-\alpha$。这就是为什么式(4.11)中给出的置信区间说成只是近似的原因。

根据上述讨论，现在引入另一个置信区间表达式。如果 X_i 是正态随机变量，随机变量 $t_n=[\overline{X}(n)-\mu]/\sqrt{S^2(n)/n}$是具有 $n-1$ 个自由度(df)的 t 分布[例子见 Hogg 和 Craig(1995，第 181-182 页)]，一个准确的(对于任意 $n\geqslant 2$)μ 的 $100(1-\alpha)\%$置信区间为：

$$\overline{X}(n)\pm t_{n-1,1-\alpha/2}\sqrt{\frac{S^2(n)}{n}} \qquad (4.12)$$

其中，$t_{n-1,1-\alpha/2}$是具有 $n-1$ 个自由度的 t 分布的上 $1-\alpha/2$ 临界点。

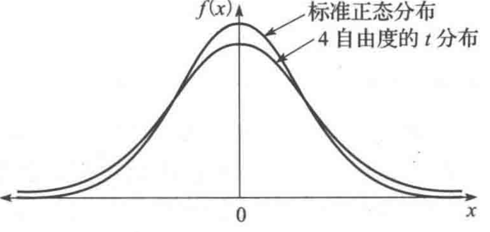


图 4.17　4 自由度的 t 分布和标准正态分布的密度函数

这些临界点在本书末尾附录中的表 A.1 中给出。4 自由度的 t 分布和标准正态分布的密度函数图如图 4.17 所示。注意，t 分布比标准正态分布具有较小的峰值和较长的尾部，所以，对于任意有限的 n，$t_{n-1,1-\alpha/2}>z_{1-\alpha/2}$。称式

(4.12)为 t 置信区间。

为了构建置信区间，式(4.12)中增减 $\overline{X}(n)$ 的量称为置信区间的半长。这是 μ 精确程度的度量。可以证明，如果将式(4.12)中的样本长度由 n 增加到 $4n$，则半长减少约 1/2(见习题 4.20)。

在实际中，X_i 的分布很少是正态的，式(4.12)给出的置信区间也是近似地覆盖。由于 $t_{n-1,1-\alpha/2}>z_{1-\alpha/2}$，式(4.12)给出的置信区间比式(4.11)给出的大，且一般覆盖度更接近所要求的水平 $1-\alpha$。由于这个原因，我们建议使用式(4.12)构建 μ 的置信区间。注意，当 $n\to+\infty$ 时，$t_{n-1,1-\alpha/2}\to z_{1-\alpha/2}$；特别地，$t_{40,0.95}$ 与 $z_{0.95}$ 的差异小于 3%。然而，在第 9，10，12 章中式(4.12)的大部分应用是小的 n，足以感觉到式(4.11)与式(4.12)之间的差别。

例 4.27 假设有 10 个观察值 1.20，1.50，1.68，1.89，0.95，1.49，1.58，1.55，0.50，1.09，是具有未知均值 μ 的正态分布，我们的目标是构建一个 μ 的 90%置信区间。从这些数据可得到：

$$\overline{X}(10)=1.34,\quad S^2(10)=0.17$$

且得到下面的 μ 的置信区间：

$$\overline{X}(10)\pm t_{9,0.95}\sqrt{\frac{S^2(10)}{10}}=1.34\pm 1.83\sqrt{\frac{0.17}{10}}=1.34\pm 0.24$$

注意用式(4.12)构建置信区间，且 $t_{9,0.95}$ 来自表 A.1。因此，根据上述解释，我们声明 μ 以 90%的置信度位于区间[1.10，1.58]。◀

现在讨论式(4.12)中给出的置信区间的覆盖度是如何受 X_i 的分布影响的。在表 4.1 中，我们给出基于 500 次独立试验的 90%置信度的估计覆盖度，每个样本长度为 $n=5$，10，20，40，每个分布是正态分布、指数分布、一自由度的 χ^2 分布(标准正态随机变量的平方；见第 6.2.2 小节即关于伽马分布的讨论)、对数正态分布(e^Y，其中 Y 是一个标准正态随机变量，见第 6.2.2 小节)，以及超指数分布，其分布函数为：

$$F(x)=0.9F_1(x)+0.1F_2(x)$$

其中，$F_1(x)$和 $F_2(x)$是指数随机变量的分布函数，其均值分别是 0.5 和 5.5。例如，指数分布且 $n=10$ 的表项按如下方式得到。从已知均值为 μ 的指数分布产生十个观察值，采用式(4.12)构建 90%置信区间，确定区间是否包含 μ(这构成一个试验)。整个过程重复 500 次，这 500 个置信区间包含 μ 的比例是 0.878。注意，因为这个表是基于 500 次实验而不是无限次实验，正态分布且 $n=10$ 的覆盖度是 0.902 而不是期望的 0.90。

表 4.1　基于 500 次试验的估计覆盖度

分布	偏度 ν	$n=5$	$n=10$	$n=20$	$n=40$
正态分布	0.00	0.910	0.902	0.898	0.900
指数分布	2.00	0.854	0.878	0.870	0.890
χ^2 分布	2.83	0.810	0.830	0.848	0.890
对数正态分布	6.18	0.758	0.768	0.842	0.852
超指数分布	6.43	0.584	0.586	0.682	0.774

观察该表，对于一个特定分布，n 越大覆盖度越接近于 0.90，这符合中心极限定理(见习题 4.22，如果实验次数更多，指数分布的结果也可能遵循这一规律)。还要注意，对于一个特定的 n，随着分布的偏度增大，覆盖度减小，其中偏度定义为

$$\nu=\frac{E[(X-\mu)^3]}{(\sigma^2)^{3/2}},\quad -\infty<\nu<+\infty$$

偏度是对称性的一个度量，对一个对称分布来说，如正态分布，偏度为 0。我们从表 4.1 得出结论，所讨论的分布偏度越大，其获得符合要求的(接近 0.90)覆盖度所需要的样本长

度越大。

*由表 4.1 可见，如果数据来自高偏度分布(例如在实践中并不罕见的对数正态分布)，则当样本长度达到 40 时，覆盖概率的退化趋势仍然很大。因此我们现在讨论一种由 Willink(2005)开发的改善型置信度，计算偏度 ν 的估计，并借此得出一个置信区间，该置信区间的覆盖度相较于式(4.12)中给出的标准 t 置信度更接近于标称值 $1-\alpha$。令：

$$\hat{\mu}_3=\frac{n\sum_{i=1}^{n}[X_i-\overline{X}(n)]^3}{(n-1)(n-2)},\quad a=\frac{\hat{\mu}_3}{6\sqrt{n}[S^2(n)]^{3/2}}$$

$$G(r)=\frac{[1+6a(r-a)^{1/3}-1]}{2a}$$

其中，$\hat{\mu}_3/[S^2(n)]^{3/2}$ 是偏度 ν 的一个估计值。则 μ 的一个近似 $100(1-\alpha)\%$ 置信区间为：

$$[\overline{X}(n)-G(t_{n-1,1-\alpha/2})\sqrt{S^2(n)/n},\overline{X}(n)-G(-t_{n-1,1-\alpha/2})\sqrt{S^2(n)/n}] \qquad (4.13)$$

*关于 Willink 置信区间的讨论可以在初次阅读时跳过。

例 4.28 对于例 4.27 中的数据，我们现在利用式(4.13)中给出的 Willink 置信区间构建一个 μ 的 90% 的置信区间。我们可知：

$$\hat{\mu}_3=-0.062,\quad a=-0.048,\quad G(r)=\frac{[1-0.288(r+0.048)]^{1/3}-1}{-0.096}$$

由此可知，μ 的 90% 的置信区间为：[1.34−0.31，1.34+0.20]或者[1.04，1.54] ◀

为了验证相较于式(4.12)中给出的 t 置信区间，利用式(4.13)中给出的 Willink 置信区间得出的覆盖概率的改善幅度，针对表 4.1 中服从对数正态分布且 $n=10$ 的记录，使用不同的随机数反复生成观察值进行试验。再次基于 500 次试验，t 置信区间和 Willink 置信区间得出的估计覆盖概率分别为 0.872 和 0.796。因此，即使是在使用高偏度的对数正态分布并且样本范围仅为 10 的情况下，Willink 置信区间生成的覆盖概率更“接近于”标准水平 0.90。另一方面，本例中 Willink 置信区间的平均半长比 t 置信区间的平均半长增加了 76%。决定使用 Willink 置信区间还是 t 置信区间，取决于覆盖概率接近于标准水平 $1-\alpha$ 及较小半长在实际问题中的相对重要性。

假设 X_1，X_2，…，X_n 是正态分布(或是近似于正态分布)的，我们要检验原假设 H_0：$\mu=\mu_0$，其中，μ_0 是 μ 的一个固定的假定值。直观上，期望如果 $|\overline{X}(n)-\mu_0|$ 很大[回顾 $\overline{X}(n)$ 是 μ 的点估计值]，H_0 不太可能为真。然而，为了设计一个具有已知统计属性的测试，需要当 H_0 为真时其分布已知的一个统计量(一个 X_i 的函数)。从上述讨论可知：如果 H_0 为真，统计量 $t_n=[\overline{X}(n)-\mu_0]/\sqrt{S^2(n)/n}$ 是一个有 $n-1$ 个自由度的 t 分布。因此，与上述直观性讨论相一致，我们的 $\mu=\mu_0$ 的(双尾)假设检验为：

$$\begin{aligned}&\text{如果 } |t_n|>t_{n-1,1-\alpha/2}, \quad \text{拒绝 } H_0\\&\text{如果 } |t_n|\leqslant t_{n-1,1-\alpha/2}, \quad \text{接受 } H_0\end{aligned} \qquad (4.14)$$

与拒绝 H_0 相对应的实线部分，即所有满足 $|x|>t_{n-1,1-\alpha/2}$ 的 x 集合称为检验的临界区，统计量 t_n 落在使 H_0 为真的临界区内的概率，显然等于 α，称为检验水平(或量)。典型地，实验者选择检验水平为 0.05 或 0.10。我们称由式(4.14)中给出的假设检验为 t 检验。

当人们进行假设检验时，可能会产生两类错误。如果当事实上 H_0 为真时拒绝 H_0，称为 I 类错误。I 类错误的概率等于水平 α，从而它是在实验者控制之下的。如果在实际上 H_0 为假时接受 H_0，称为 II 类错误。对于固定的水平 α 和样本长度 n，II 类错误的概率，我们用 β 表示，取决于什么是确实为真(相比于 H_0：$\mu=\mu_0$)，且可能是未知的。我们称 $\delta=1-\beta$ 为检验的能力，且它等于当 H_0 为假时拒绝 H_0 的概率(显然，希望检验能力高)。如果 α 是固定的，检验的能力只能随着 n 的增加而增加。由于检验能力可能很低且对于我们是未知的，当统计量 t_n 没有位于临界区域时，今后将称我们“拒绝 H_0 失败”(而不是

“接受 H_0”，当 H_0 没有被拒绝，我们通常没有任何把握知道 H_0 是真还是假，因为我们的检验也许并没有足够的能力来发现 H_0 和其实际为真之间的区别）。

例 4.29 对于例 4.27 的数据，假设想要检测 $\mu=1$ 在水平 $\alpha=0.10$ 的原假设 H_0。由于

$$t_{10}=\frac{\overline{X}(10)-1}{\sqrt{S^2(10)/10}}=\frac{0.34}{\sqrt{0.17/10}}=2.65>1.83=t_{9,0.95}$$

我们拒绝 H_0。◀

例 4.30 对于例 4.29 中 $\mu=1$ 时的原假设 H_0，事实上，当 X_i 是均值、$\mu=1.5$ 和标准差 $\sigma=1$ 的正态分布时，我们可以估计检验能力。在假定 $\mu=1.5$ 和 $\sigma=1$ 条件下随机产生统计量 $t_{10}=[\overline{X}(10)-1]/\sqrt{S^2(10)/10}$ 的 1 000 个独立观察值（当然，X_i 是正态分布的）。在 1 000 个观察值中有 433 个观察值，$|t_{10}|>1.83$，因此，估计能力是 $\hat{\delta}=0.433$。因此，如果 $\mu=1.5$ 和 $\sigma=1$，我们在检验水平 $\alpha=0.10$，大约只有 43% 的次数会拒绝原假设 $\mu=1$。为了观察标准差 σ 对检验能力的影响，我们在 $\mu=1.5$、$\sigma=0.75$ 时也产生 t_{10} 的 1 000 个观察值和 $\mu=1.5$、$\sigma=0.5$ 时 t_{10} 的 1 000 个观察值（所有 X_i 都是正态分布的）。估计的能力分别是 $\hat{\delta}=0.619$ 和 $\hat{\delta}=0.900$。显然能力是 σ 的减函数，这一点并不令人奇怪，因为当 σ 较小时，我们会期望更好地区分真实的均值 1.5 和假设均值 1[注意，在正态采样的情况下，像本例，不考虑仿真的需要，检验能力实际上可以精确计算出来；关于非中心 t 分布，请参阅关于统计学的高级教材，例如 Bickel 和 Doksum(2000)]。◀

应该指出的是，由式(4.21)给出的置信区间和由式(4.14)给出的假设检验之间存在密切关系。特别是，假设检验和置信区间假定两者具有相同的 α 值时，拒绝原假设 $H_0(\mu=\mu_0)$ 与等价于 μ_0 没有包含在 μ 的置信区间里。

4.6　强大数定律

可以认为，概率理论中第二个最重要的结论（继中心极限定理之后）就是强大数定律。令 X_1，X_2，…，X_n 为具有有限均值 μ 的独立同分布随机变量。则强大数定律如下[证明见 Chung(1974，第 126 页)]。

定理 4.2　当 $n\to+\infty$ 时，$\overline{X}(n)$ 以概率 1 趋向于 μ。

该定理说，实际上，如果执行无限次实验，每次都得到一个 $\overline{X}(n)$，且 n 足够大，则几乎对所有的实验来说，$\overline{X}(n)$ 将任意程度地接近 μ。

例 4.31 假设 X_1，X_2，…，X_n 是 $\mu=1$ 与 $\sigma^2=0.01$ 的独立同分布的正态随机变量。图 4.18 绘出了从该分布中采样得到的不同 n 的 $\overline{X}(n)$ 值。注意，当 $n\geqslant 28$ 时，$\overline{X}(n)$ 与 μ 的差异小于 1%。◀

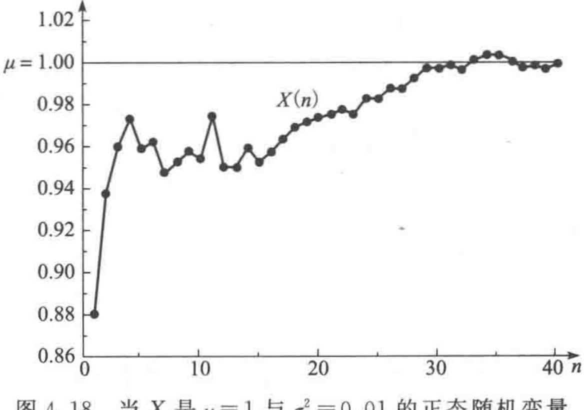


图 4.18　当 X_i 是 $\mu=1$ 与 $\sigma^2=0.01$ 的正态随机变量时不同 n 的 $\overline{X}(n)$ 值

4.7　用均值来替代概率分布的危险性

仿真分析人员有时在其仿真模型中用输入的均值来代替其概率分布。这个做法可能是由于分析人员缺乏理解或者缺乏实际分布形式的信息（例如，只有分布均值的估计值可用）而引起的。下面的例子说明了这种做法的危险性。

例 4.32 考虑由一台机床构成的制造系统。假设“未加工的”零件以均值为 1min 的指数分布的间隔时间的机床，在该机床的加工时间是均值为 0.99min 的指数分布。因此，这个系统是一个 $M/M/1$ 排队系统，利用率 $\rho=0.99$。此外，可以证明，在一次长时间运行中，一个零件在队列中的平均延误是 98.01min[见附录 1B 或 Gross 和 Harris(1998，第 67 页)]。另一方面，如果我们用相对应的均值替代每一个分布(例如，如果顾客在 1min 时刻，2min 时刻，…到达，并且如果每个零件的加工时间正好是 0.99min)，则没有零件会在队列中延误。一般情况下，输入分布的方差和均值影响排队系统的输出性能度量，正如在附录 1B 最后指出的那样。◀

附录 4A　协方差平稳过程的说明

考虑在 0 时刻没有顾客出现的 $M/M/1$ 队列的过程$\{D_i,\ i\geqslant1\}$。显然，$D_1=0$，但是对于 $i=2,\ 3,\ \cdots$，$P(D_i>0)>0$。因此，$E(D_1)=0$，且对于 $i=2,\ 3,\ \cdots$，$E(D_i)>0$，这意味着$\{D_i,\ i\geqslant1\}$不是协方差平稳的。然而，如果 $\rho<1$，可以证明对于所有 $x\geqslant0$，有：

$$P(D_i\leqslant x)\rightarrow(1-\rho)+\rho(1-e^{-(\omega-\lambda)x}),\quad 当\ i\rightarrow+\infty \tag{4.15}$$

由式(4.15)和例 4.19 中 D_{i+1} 的等式可以得出，如果从 D_1，D_2，…中删除前 k 个观察值，且 k 足够大，则过程 D_{k+1}，D_{k+2}，…是(近似)协方差平稳的。因此，当“考虑协方差平稳的 $M/M/1$ 队列的过程$\{D_i,\ i\geqslant1\}$”，旨在观察第一个延误之前让 $M/M/1$ 队列“预热”一段时间。

考虑例 4.23 中当 $I_1=S$ 时的库存系统的过程$\{C_i,\ i\geqslant1\}$。由于对于 $i=2,\ 3,\ \cdots$，$P(I_i=S)\neq1$，可以得出$\{C_i,\ i\geqslant1\}$不是协方差平稳的。然而，可以证明当 $i\rightarrow+\infty$ 时，$P(C_i\leqslant x)$收敛于一个有限分布函数[见 Wagner(1969，第 A48 页)]。因此，对于足够大的 k，C_{k+1}，C_{k+2}，…是(近似)协方差平稳的。此外，图 4.11 所示的是在观察第一次成本前，库存系统预热一段时间的相关系数图。

习题

4.1 假设 X 是一个离散随机变量，其概率质量函数为 $p(1)=\frac{1}{10}$，$p(2)=\frac{3}{10}$，$p(3)=\frac{2}{10}$，$p(4)=\frac{3}{10}$和 $p(5)=\frac{1}{10}$。

(a) 绘出 $p(x)$图。

(b) 计算 $F(x)$并绘图。

(c) 计算 $P(1.4\leqslant X\leqslant4.2)$，$E(X)$和$\text{var}(X)$。

4.2 假设 X 是一个连续随机变量，概率密度函数为：

$$f(x)=x^2+\frac{2}{3}x+\frac{1}{3},\quad 0\leqslant x\leqslant c$$

(a) c 的值必须是什么？

假定 c 是这个值，进行下面的事情：

(b) 绘出 $f(x)$图。

(c) 计算 $F(x)$并绘图。

(d) 计算 $P\left(\frac{1}{3}\leqslant x\leqslant\frac{2}{3}\right)$，$E(X)$和$\text{var}(X)$。

4.3 假设 X 和 Y 是联合离散随机变量，有：

$$p(x,y)=\begin{cases}\dfrac{2}{n(n+1)}, & x=1,2,\cdots,n\ 且\ y=1,2,\cdots,x\\ 0, & 其他\end{cases}$$

计算 $p_X(x)$和 $p_Y(y)$并确定 X 和 Y 是否独立。

4.4 假设 X 和 Y 是联合离散随机变量，有：

$$p(x,y)=\begin{cases}\dfrac{x+y}{30}, & x=0,1,2 \text{ 且 } y=1,2,3\\ 0, & \text{其他}\end{cases}$$

(a) 计算 $p_X(x)$和 $p_Y(y)$并绘图。

(b) X 和 Y 是独立的吗?

(c) 计算 $F_X(x)$和 $F_Y(y)$并绘图。

(d) 计算 $E(X)$、$\text{var}(X)$、$E(Y)$、$\text{var}(Y)$、$\text{cov}(X, Y)$和 $\text{cor}(X, Y)$。

4.5 例 4.10 中，如果两张卡片的采样是有放回的，随机变量 X 和 Y 是独立的吗?

4.6 假设 X 和 Y 是联合连续随机变量，有：

$$f(x,y)=\begin{cases}32x^3y^7, & 0<x<1 \text{ 且 } 1<y<2\\ 0, & \text{其他}\end{cases}$$

计算 $f_X(x)$和 $f_Y(y)$并确定 X 和 Y 是否独立。

4.7 假设 X 和 Y 是联合连续随机变量，有：

$$f(x,y)=\begin{cases}y-x, & 0<x<1 \text{ 且 } 1<y<2\\ 0, & \text{其他}\end{cases}$$

(a) 计算 $f_X(x)$和 $f_Y(y)$并绘图。

(b) X 和 Y 是独立的吗?

(c) 计算 $F_X(x)$和 $F_Y(y)$。

(d) 计算 $E(X)$、$\text{var}(X)$、$E(Y)$、$\text{var}(Y)$、$\text{cov}(X, Y)$和 $\text{cor}(X, Y)$。

4.8 如果 X 和 Y 是联合连续随机变量，有联合概率密度函数 $f(x, y)$，且 X 和 Y 是独立的，证明 $\text{cov}(X, Y)=0$。从而，X 和 Y 是独立的，意味着$E(XY)=E(X)E(Y)$。

4.9 假设 X 是一个离散随机变量，对于 $x=-2$，-1，1，2，有 $p_X(x)=0.25$。令 Y 也是一个离散随机变量，且满足 $Y=X^2$。显然，X 和 Y 不是独立的。但是，请证明 $\text{cov}(X, Y)=0$。从而，不相关的随机变量不一定是独立的。

4.10 假设 X_1 和 X_2 是联合正态分布随机变量，其联合概率密度函数为：

$$f_{X_1,X_2}(x_1,x_2)=\frac{1}{2\pi\sqrt{\sigma_1^2\sigma_2^2(1-\rho_{12}^2)}}e^{-q/2}, \quad -\infty<x_1<+\infty \text{ 且 } -\infty<x_2<+\infty$$

其中，

$$q=\frac{1}{1-\rho_{12}^2}\left[\frac{(x_1-\mu_1)^2}{\sigma_1^2}-2\rho_{12}\frac{(x_1-\mu_1)(x_2-\mu_2)}{\sqrt{\sigma_1^2\sigma_2^2}}+\frac{(x_2-\mu_2)^2}{\sigma_2^2}\right]$$

如果 $\rho_{12}=0$，证明 X_1 和 X_2 是独立的。

4.11 假设 X 和 Y 是随机变量，并有 $Y=aX+b$，且 a 和 b 是常数。证明

$$\text{cor}(X,Y)=\begin{cases}+1, & a>0\\ -1, & a<0\end{cases}$$

这就是为什么称相关性是线性依赖的一个度量。

4.12 若 X_1 和 X_2 是随机变量，则由施瓦兹(Schwarz)不等式有 $E(X_1^2)E(X_2^2)\geqslant[E(X_1X_2)]^2$。应用这个事实来证明$-1\leqslant\rho_{12}\leqslant1$。

4.13 对于任意随机变量 X_1、X_2 以及任意数 a_1、a_2，证明

$$\text{var}(a_1X_1+a_2X_2)=a_1^2\text{var}(X_1)+2a_1a_2\text{cov}(X_1,X_2)+a_2^2\text{var}(X_2)$$

4.14 证明例 4.19 中的 D_{i+1}的等式。

4.15 利用例 4.19 中 D_{i+1}的等式，编写一个大约 15 行的 C 程序代码来仿真一个平均到达时间间隔为 1，平均服务时间为 0.5 的 $M/M/1$ 队列。运行程序直到观察到 1 000 组 D_i 值，并计算 $\overline{D}(1\,000)$。程序不需要仿真时钟、事件表或者定时程序。

4.16 对于任意随机变量 X_1，X_2，…，X_n 和任意数 a_1，a_2，…，a_n 都有 $E\left(\sum_{i=1}^{n}a_iX_i\right)=\sum_{i=1}^{n}a_iE(X_i)$，利用该事实，证明如果 X_1，X_2，…，X_n 是均值为 μ，方差为 σ^2 的独立同分布随机变量，则 $E[\overline{X}(n)]=\mu$ 且$E[S^2(n)]=\sigma^2$。证明如果 X_i 是相关的，第一个结果仍然成立。

4.17 证明等式(4.7)是正确的。

4.18 若 X_1，X_2，…，X_n 是独立同分布随机变量，均值为 μ，方差为 σ^2，则计算$\text{cov}[\overline{X}(n),S^2(n)]$。什么时候这个协方差等于 0?

4.19 证明等式(4.10)中的两个概率说明在取等式时是正确的。

4.20 对于式(4.12)给出的置信区间，证明如果我们将采样长度从 n 增加到 $4n$，则半长减少 1/2。

4.21 解释为什么例 4.26 中 90%置信区间只包含 10 个观察值中的 5 个。

4.22 对于由式(4.12)给出的置信区间，证明当 $n \to +\infty$ 时覆盖趋向于 $1-\alpha$。

4.23 假设 7.3，6.1，3.8，8.4，6.9，7.1，5.3，8.2，4.9，5.8 是一个具有未知均值 μ 的分布(偏度不高)的 10 个观察值。计算 $\overline{X}(10)$，$S^2(10)$和 μ 的约 95%的置信区间。

4.24 对于习题 4.23 中的数据，检验原假设 H_0：$\mu=6$，水平 $\alpha=0.05$。

4.25 解释例 4.30 中，为什么当 n 变大时，检验能力会趋近于 1。(提示：当 n 变大时，统计量 $t_n=[\overline{X}(n)-1]/\sqrt{S^2(n)/n}$中的分子和分母会发生何种变化)

4.26 例 4.30 中，$n=10$ 时估计能力为 0.433 且备择假设 H_1：$\mu=15$，$\sigma=1$。

(a) 当 $n=10$ 时，如果备择假设变为 H_1：$\mu=1.25$，$\sigma=1$，则能力是会增加还是会下降？利用一个适当的正态分布随机采样(例如，Excel)产生的数据证明您的答案。

(b) 当 $n=10$ 时，如果备择假设变为 H_1：$\mu=1.5$，$\sigma=0.75$，则能力是会增加还是会下降？利用一个适当的正态分布随机采样(例如，在 Excel)产生的数据证明您的答案。

4.27 假设有一个制造过程，要生产直径为 0.5 英寸的轴承。公司检验 $n=50$ 个轴承并发现 $\overline{X}(50)=0.45$，$S^2(n)=0.06$。测试原假设 H_0：$\mu=0.05$ 以及备择假设 H_1：$\mu\neq0.05$，其中显著性水平为 $\alpha=0.05$。同时，构造 μ 的置信度为 95%的置信区间。

4.28 从代数的角度证明利用式(4.14)中给出的 t 检验拒绝原假设 H_0：$\mu=\mu_0$ 等价于利用式(4.12)中给出的不包含 μ_0 的 t 置信区间该原假设。

4.29 假设 X 和 Y 是随机变量，具有未知协方差 cov(X，Y)。若 X_i，Y_i对于(对于 $i=1$，2，…，n)是 X，Y 的独立观察值，证明

$$\widehat{\mathrm{cov}}(X,Y)=\frac{\sum_{i=1}^{n}[X_i-\overline{X}(n)][Y_i-\overline{Y}(n)]}{n-1}$$

是 cov(X，Y)的一个无偏估计值。

4.30 称随机变量 X 具有无记忆性，若

$$P(X>t+s\,|\,X>t)=P(X>s),\quad 对于所有\ t,s>0$$

证明指数分布具有无记忆性，[条件概率 $P(X>t+s\,|\,X>t)$是在给定事件$\{X>t\}$已经发生的条件下，事件$\{X>t+s\}$发生的概率；见 Ross(2003，第 3 章)]

4.31 一个具有参数 $p(0<p<1)$的几何分布，其概率质量函数为：

$$p(x)=p(1-p)^x,\quad 对于\ x=0,1,2,\cdots$$

证明这个分布具有无记忆性。

4.32 假设一个人有 k 把钥匙，一把钥匙打开一扇门。计算在以下两种情况下打开门所需钥匙的期望数目。

(a) 无放回地一次试一把钥匙。

(b) 有放回地一次试一把钥匙。(提示：以第一次尝试的结果为条件)

4.33 均值、中值和众数对于每个对称分布都是相等的吗？

4.34 在例 4.32 中当间隔时间和加工时间均为指数分布或者均为常数时，长时间运行的吞吐量(离开率)是否会变大？

第5章 建立有效、可信、适度详细的仿真模型

5.1 引言及定义

仿真分析人员所面对的最困难的问题之一就是企图确定一个仿真模型是否准确地表示了被研究的系统，即模型的是否有效。本章我们就如何建立一个有效、可信的模型进行实用的讨论。我们还给出如何界定复杂系统模型的详细程度的准则，也是一个关键且有挑战性的问题。本章内容参考了很多已有文献资料、Averill M. Law & Associates 的咨询研究，以及 1977 年以来参加作者的仿真短期课程的成千上万人的经验。我们给出超过 40 个例子来说明所提出的概念。

关于确认和校验的重要工作的有 Balci(1998)、Banks 等(2005)、Carson(1986，2002)、Feltner 与 Weiner(1985)、Law(2000，2005)、Naylor 与 Finger(1967)、Sargent(2004)、Shannon(1975)和 Van Horn(1971)。对现有仿真模型评价的参考文献包括 Fossett et al.(1991)、Gass(1983)、Gass 与 Thompson(1980)和 Knepell 与 Arangno(1993)。

我们从定义本章中用到的重要术语开始，包括校验、确认和可信性。校验是关于确定“假设文档”(见第 5.4.3 小节)是否已经正确地翻译成计算机“程序”，即调试仿真计算机程序。虽然校验在概念上简单，由于存在大量的潜在逻辑路径，调试一个大型仿真程序是一项困难且艰苦的任务。校验仿真计算机程序的技术在第 5.3 节中讨论。

确认是确定一个仿真模型对研究的特定目标来说是否是系统的准确表示的过程[Fishman and Kiviat(1968)看来是第一个给出类似这样定义的人]。下面是关于确认的一般观点。

- 概念上，如果仿真模型是“有效的”，则它可以用于对系统做决策，即如果模型对于用系统本身做实验来说是可行的且成本低的话，则该系统与模型类似。
- 确认过程的难易程度取决于被建模系统的复杂性和当前系统是否存在(见第 5.4.5 小节)。例如，一个街道银行模型的确认是相当简单的，因为它可以近距离考察。然而，一个 2025 年海军武器系统模型的有效性不可能完全确认，因为战役发生的位置和敌军武器的性质是未知的。
- 无论在建模上付出多少努力，一个复杂系统的仿真模型只能是真实系统的一个近似。不存在模型绝对有效这回事，甚至也从未有人这样期望过。一般来说，在模型开发中投入的时间越多(因而费用就越多)，模型应该越有效。然而，最有效的模型不一定是最符合成本效益的。例如，由于可能需要大量数据收集，增加模型的有效性超过了一定水平可能是相当昂贵的，但是可能并未带来认识或决策的显著改善。
- 仿真模型应该总是针对一组特定目标开发的，对一种目标有效的模型也许对另一种目标无效。
- 用于确认模型的性能度量应该包括决策者实际用于评估系统设计的那些内容。
- 确认不是在仿真模型开发以后要做的事，除非有足够的时间和金钱。不幸的是，我们的经验表明，这一建议往往没有遵循。

例 5.1 一个组织付给一个咨询公司 $500 000 来进行一项“仿真研究”。在研究被认为完成后，客户组织的一个人打电话并询问：“你能不能在五分钟内通过电话告诉我是如

何确认我的模型的?”　◀

- 每当考虑将仿真模型用于一个新的应用时，它的有效性应该重新审查。当前的目的可能与原来的目的有着很大的不同，或者时间的推移可能使得某些模型参数变得无效。

如果管理者和其他关键项目人员认可仿真模型和它的结果是“正确的”，则他们具有可信性(我们此后用术语“管理者”指代适合上下文的管理者，决策者，或者顾客)。注意一个可信的模型不一定是有效的，反之亦然。而且，模型可以是可信的，但实际上没有用做决策过程的帮助工具。例如，模型可能是可信的但由于政治或经济的原因并没有使用。下面几点有助于确定模型的可信性。

- 管理者理解和同意模型的假设(见第 5.4.2 小节)。
- 模型确认和校验的演示。
- 管理者拥有和参与项目。
- 模型开发者的信誉。
- 栩栩如生的动画。

美国国防部(DoD)是仿真模型的一大用户，近些年一直对校验、确认和称为认证的概念方面(VV&A)给予了很大关注。认证[见国防建模与仿真办公室(2000，2003)]是仿真模型对特定目的是可接受的官方认定(由项目赞助商)。委托 DoD 内部来完成认证的主要原因是，必须有人对某一特定应用使用一个模型的决策负责，因为可能要花费大量的金钱以及危及人民的生命。而且，大量的军事分析使用以前遗留下来的模型，这些模型可能是为另一个应用或者由另一个组织开发的。在认证决策中考虑的问题包括：

- 已完成的校验和确认；
- 模型的可信性；
- 仿真模型开发和使用的历史(例如，模型开发者和类似的应用)；
- 可获得数据的质量；
- 文档的质量；
- 仿真模型已知的问题或者局限性。

确认、校验和确定可信性的时段和关系如图 5.1 所示。矩形表示模型或者相关系统的状态，实线水平箭头对应从一个状态移动到另一个必需的动作，虚曲线箭头表示三个主要的概念最显著出现的地方。每个实线箭头下方的数字对应于一次有效仿真研究的各步，如第 1.7 节中讨论的那样。我们并不计划阐释图中的反馈弧。

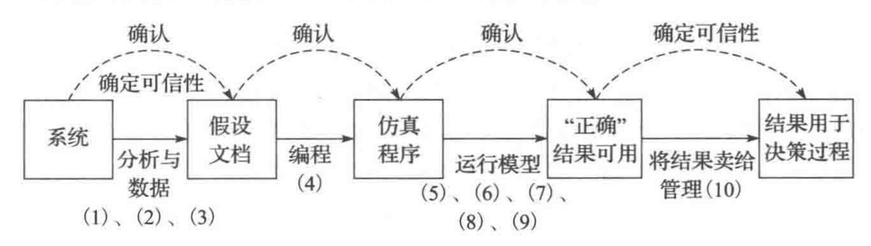


图 5.1　确认、校验与确定可信性的时段和关系

应该将确认与输出分析(第 9 章到第 12 章的主题)进行对比，输出分析是有关估计一个仿真模型的(不一定是系统的)性能的真实度量的统计学问题。有关输出分析的主题包括仿真运行长度、预热期长度(如果有的话)、模型(采用不同随机数)独立运行的次数。

为了更好地理解确认和输出分析之间的区别，假设我们想要估计某系统的均值 μ_S。假设我们建立了一个仿真模型其对应的均值是 μ_M。我们执行一次仿真运行并得到 μ_M 的一个估计值 $\hat{\mu}_M$。则用 $\hat{\mu}_M$ 作为 μ_M 估计值的误差为：

$$\hat{\mu}\text{的误差} = |\hat{\mu}_M - \mu_S|$$

$$= |\hat{\mu}_M - \mu_M + \mu_M - \mu_S|$$
$$\leqslant |\hat{\mu}_M - \mu_M| + |\mu_M - \mu_S| \quad \text{(根据三角不等式)}$$

确认关注的是使第二个绝对值小(在上一行中)，而输出分析关注的是使第一个绝对值小。因此，为得到系统均值的一个好的估计，我们必须既关注确认也关注输出分析。

5.2 确定模型详细程度的准则

一个专业的仿真人员必须确定一个复杂真实世界系统的哪些方面确实需要纳入仿真模型，以及达到何种详细程度，哪些方面可以安全地忽略掉。很少需要将系统中每一个元素与模型中的每一个元素一一对应。为做出一个有效的决策，很少需要对系统的每个方面建模，且这样做可能导致过长的模型运行时间，错过了最后期限，或者对重要的系统因素不能给以足够的重视。

例 5.2 一个狗粮生产商让咨询公司建立其生产线的仿真模型，这条生产线每天以恒定的速度生产一百万罐。由于每罐食品在模型中用一个独立的实体表示，这个模型运行代价很高，因此并不实用。几年后这个模型被重写，将生产过程处理成一个“连续流”(见第 1.2 节)。新的模型得出了精确的结果且执行时间只是原模型所必需的一小部分。 ◀

例 5.3 一个 1.5 英里长的工厂的仿真模型是在 1985 年建立的，花费了 \$250 000。然而，由于该模型过于详细，以至于计算机内存要求太高而导致从来没有运行过。 ◀

我们现在给出一些确定仿真模型所需详细程度的某些一般准则[也可见 Law(1991)and Robinson(2004，第 87-92 页)]。

- 认真定义要研究的问题的细节以及将用于评估的性能度量。模型的有效性不是万能的，而是为特定目的而设计的。如果关注的问题还没有搞清楚，那么确定合适的模型详细程度是不可能的。由于某些模型只可以精确地评估性能的一个指标而不能精确评估另一个，详细说明关注的性能度量也很重要。例如，一个制造系统的简单模型可以准确预测吞吐量(例如，每天生产的零件)，但是不能确定在制品所需要的占地面积(见例 13.3)。最后，理解管理者的需要非常重要。对错误问题再好的模型也是不会被使用的。问题形式化通常在最初阶段的启动会完成，参会人员给出该系统的所有关键方面的描述。

例 5.4 美军军事分析人员在一个仿真模型上工作了 6 个月而没有与需要这个仿真模型的将军沟通。在五角大楼对这项研究的简介中，将军在陈述 5 分钟后退席了：“这个问题我不感兴趣”。 ◀

- 实体通过仿真模型并不总是必须与实体通过对应系统相同(见例 5.5)。此外，并不总是必须要对系统的每个组件完全详细地建模(见例 5.26)

例 5.5 一个大食品生产商建立其糕点生产线的仿真模型。最初，他们试图把每块饼干建模成独立实体，但该模型的计算量的要求使得这个方法行不通。因此，公司被迫使用一箱饼干作为实体通过模型。通过使用灵敏度分析确定了该建模方法的有效性(见下文和例 5.25)。 ◀

- 使用主题专家(SME)和灵敏度分析来帮助决定模型的详细程度。熟悉系统的人与关注系统的人同样重要，要问他们，所建议系统的组件哪些可能是最重要的，从而必须仔细地建模。可利用灵敏度分析(见第 5.4.4 小节)来确定哪些系统因素(例如参数或分布)对所要求的性能度量有最大的影响。在给定的有限模型开发时间，人们显然应该集中于最重要的因素。
- 建模初学者经常犯的一个错误是模型细节过多。因此，我们建议从一个“适度详细的”模型开始，如果需要的话可以随后再完善它。模型的特定版本的适用性部分地是通过将模型提交给领域专家和管理者来决定的。定期与这些人沟通也可以保持他

们对仿真研究的关心。

例5.6 我们开发一个宠物食品制造系统的仿真模型，由肉类加工厂和一个罐头厂组成。在肉类加工厂中，肉要么被绞碎，要么切成块，随后放入桶中，通过高架传送带送往罐头厂。在罐头厂中，桶被倾倒到搅拌器中，在里面加工肉类随后将其送到填装机/封罐机来罐装。空桶送回肉类加工厂重装。起初，决定生产块状产品的系统相对不重要，因此使用了一种简单的方法来建模。然而，在模型结构化的预演中(见第5.4.3小节)，机器操作人员说这个子系统实际上更加复杂。为获得这些项目团队的成员的信任，我们必须包含机器故障和资源竞争。此外，在最初的模型运行之后，必须按照搅拌器操作者的建议对模型做出一些其他的改变。◀

- 在模型中不要有比解决关注的问题所必需的更多细节，前提是，模型必须足够详细保证其是可信的。因此，出于可信性的考虑，有时模型中必须有某些对模型确认严格来说并不需要的东西。
- 模型的详细程度应该与可用数据的类型相一致。用于设计新的制造系统的模型的详细程度通常会比调整一个已有系统的模型小一些，因为对建议的系统来说可用的数据会很少甚至没有。
- 在实际的所有仿真研究中，时间和经费约束是确定模型详细程度的主要因素。
- 如果要研究的因素(关注的方面)数量多，则使用一个“粗”仿真模型或者一个解析模型来识别哪些因素对于系统性能有重大影响。随后，着重这些因素，建立一个“详细”仿真模型[例子见 Haider，Noller 与 Robey(1986)]。注意，在诸如制造系统和通信网络这些应用领域，有商业软件包可用于性能解析分析。统计实验设计(见第12章)对于确定重要因素也是有用的。

5.3 仿真计算机程序校验

在这节中我们讨论可以用于调试仿真模型的计算机程序的8项技术[软件工程领域以外的技术见 Balci(1998)]。这些技术中的某些可以用于调试任何计算机程序，另外一些我们认为是针对仿真建模的。

技术1

在开发仿真模型的过程中，编写与调试在模块或者子程序中的计算机仿真程序。举例来说，对于一个10 000条语句的仿真模型，在写完整个程序之前不进行任何调试是很坏的编程习惯。当最终运行这个很大的且未经过测试的程序时，它几乎必然不会执行，确定该程序中错误的位置将是非常困难的。相反地应该首先编写和调试仿真模型的主程序和少数几个关键子程序，也许把其他需要的子程序表示为“伪程序”或“存根”。之后，应该添加其他的子程序或者细节的层次并调试成功，直到开发出一个能够令人满意地表示所研究的系统的模型。一般来说，从一个“适度详细”的模型开始，逐步达到需要的复杂程度，比“直接”开发一个复杂的模型总会好一些，后者会比需要的模型更详细且运行会过于昂贵(进一步讨论见例5.25)。

例5.7 对于第2.6节中讨论的可以换队的多出纳台银行例子，一个好的编程方案是首先编写和调试不让顾客队列间换队的计算机程序。◀

技术2

在开发大型仿真模型时，安排不止一个人检查计算机程序是明智的，因为一个特定子程序的编写者可能陷入一种思维定式，因此，也许不是一个好的评论人员。在一些组织中，这种思想正规地得到执行并称为结构化代码走查。例如，所有建模团队的成员，譬如说，系统分析者、编程人员，等等，聚集到一个房间里，给每人发一组待调试的特定子程序的拷贝。然后该子程序的开发者一句一句地遍历整个程序，但在每个人都确信这条语句是正确的之后才推进到下一条语句。

技术 3

在多种设定的输入参数下运行仿真，并检查输出的合理性。在某些情况下，可以精确地计算一些简单的性能度量，并用于比较(例子学习见第 13.6 节)。

例 5.8 对于多个有 s 个服务台并行的排队系统，可以证明，服务台长运行的平均利用率是 $\rho=\lambda/(s\omega)$(符号见附录 1B)。因此，如果一次仿真运行的平均利用率接近利用率因子 ρ，这表明该程序可能运行正确。◀

技术 4

可用于调试一个离散事件仿真程序的最强有力的技术之一是“跟踪”。在一次跟踪中，被仿真系统的状态，即事件表的内容、状态变量、某些统计计数器，等等，在每个事件发生后马上显示并与手工计算结果比较来判断程序是否按照预期运行。在进行一次跟踪中最好是评估每一个可能的程序路径以及程序处理“极端”情况的能力。有时这样一次遍历检测可能需要为这个模型准备的专门的(可能是确定的)输入数据。大部分仿真软件包提供跟踪的能力。

批处理模式的跟踪往往产生大量的输出，必须逐个事件地检查错误。遗憾的是，一些关键信息(分析者未要求的)可能在跟踪中遗漏掉；或者，更糟糕的是，一个特定错误在“短暂的”调试仿真运行过程中可能没有发生。这两个困难中任何一个都需要仿真重新运行。因此人们通常愿意用交互式调试器来发现程序错误。

交互式调试器允许分析者在选定的时间点上停止仿真，检查或者可能改变某些变量的值。后一种能力可以用于“迫使”某些类型错误发生。许多现代仿真软件包都有交互式调试器。

例 5.9 表 5.1 给出了第 1.4.2 小节中的对单服务台队列的直观解释的跟踪。表的第一行是在 0 时刻刚刚完成初始化的系统的快照，第二行是系统刚刚发生第一个事件(一个到达)后的快照，等等。◀

表 5.1　1.4.2 小节中的单服务台队列的部分跟踪

事件	时刻	服务台状态	在队人数	到达时间	事件表		被延误的顾客数	总延误	在队人数函数的面积	忙函数的面积
					到达	离去				
初始化	0	0	0		0.4	∞	0	0	0	0
到达	0.4	1	0		1.6	2.4	1	0	0	0
到达	1.6	1	1	1.6	2.1	2.4	1	0	0	1.2
到达	2.1	1	2	1.6，2.1	3.8	2.4	1	0	0.5	1.7
离去	2.4	1	1	2.1	3.8	3.1	2	0.8	1.1	2.0
离去	3.1	1	0		3.8	3.3	3	0.8	1.8	2.7
离去	3.3	0	0		3.8	∞	3	0.8	1.8	2.9

技术 5

如果可能的话，应该在模型真实特征已知或者易于计算的简化假设下运行模型。

例 5.10 对于第 2.7 节中的加工车间模型，不可能解析地计算期望的系统特性。因此，必须采用仿真。为了调试仿真模型，我们可以首先运行第 2.7.2 小节中的通用模型，它只有一个工作站，该工作站有一台机器，且只有一类作业(到达速率为 0.3/0.25=1.2 个作业/小时)。得到的模型称为 $M/E_2/1$ 队列，且具有已知的瞬态和稳态特性[见 Kelton (1985)、Gross 等(2009)]。表 5.2 给出了队列中稳态平均数，平均利用率、队列中平均延误时间的理论值，以及从一次长度为 2 000 个 8 小时工作日的仿真运行中得出的这些量的估计值。由于估计值非常接近真实值，我们在某种程度上相信计算机程序是正确的。◀

表 5.2　一个简化的加工车间模型($M/E_2/1$ 队列)
理论值(T)和仿真估计值(S)

队列中平均数		平均利用率		队列中平均延误时间	
T	S	T	S	T	S
0.676	0.685	0.600	0.604	0.563	0.565

该程序的更加确定的测试可以通过运行第 2.7.2 小节中的通用模型得到，初始工作站数(5)，每个工作站上初始机器数(3，2，4，3，1)，只有一类作业，且在每个工作站上是指数服务时间(与相应的 2-厄兰分布的服务时间具有相同的均值)。在效果上，由此产生的模型是串联的 4 个多服务台队列，第一个队列是 $M/M/4$，第二个队列是 $M/M/3$，等等[长期运行的 $M/M/s$ 队列(s 是服务台数目)的离去间隔时间是独立同分布指数随机变量；见 Gross 等(2009)]。此外，$M/M/s$ 队列的稳态特性是已知的[见 Gross 等(2009)]。对于每一个工作站，表 5.3 给出了其队列中稳态平均数，平均利用率和队列中平均延误时间的理论值，以及从一次长度为 2 000 个 8 小时工作日的仿真运行中得出的这些量的估计值。仿真估计值又一次相当接近理论值，这使得程序更加可信。

表 5.3　一个简化的加工车间模型(四个串行的多服务台队列)
理论值(T)和仿真估计值(S)

工作站	队列中平均数		平均利用率		队列中平均延误时间	
	T	S	T	S	T	S
3	0.001	0.001	0.150	0.149	0.001	0.001
1	0.012	0.012	0.240	0.238	0.010	0.010
2	0.359	0.350	0.510	0.508	0.299	0.292
5	0.900	0.902	0.600	0.601	0.750	0.752

例 5.11 我们开发一个移动电话服务大供应商的仿真模型，目标是确定一些服务备选网络配置的长期有效性(可用时间的比例)。最初，我们试图通过解析的方法计算有效性，如连续时间马尔可夫链和条件期望[举例见 Ross(2003)]，但是我们只能获得简单案例的结果。因此，我们需要采用仿真并且我们通过比较在简单案例中的仿真和解析结果来部分校验我们的仿真模型。◀

技术 6

对于某些类型的仿真模型，观察仿真输出的动画往往是很有帮助的(见第 3.4.3 小节)。

例 5.12 汽车交通路口网络的仿真模型已经开发出来，假设要进行调试，以用于某些时候研究各种信号灯排序策略这类问题。然而，当仿真的交通流生成动画时，发现被仿真的车辆实际上在路口相撞了；随后的计算机程序检测发现了许多之前未检测出来的错误。◀

技术 7

计算每一个仿真输入概率分布的样本均值和样本方差，并与期望的(例如，历史上的)均值和方差进行比较。这能够说明，这些值是从这些分布中正确地生成的。

例 5.13 伽马和韦布尔分布的参数在不同的仿真软件包和书中定义不同。因此，这项技术在这时就很有价值。◀

技术 8

采用商用仿真软件包来减少需要的编程量。另一方面，当使用仿真软件包(特别是最近发布的)时必须要小心，因为其中可能包含一些细微的错误。而且，仿真软件包有强大的高层宏语句，有时并没有很好地文档化。

5.4 提高模型有效性和可信性的方法

本节讨论六类提高仿真模型有效性和可信性的方法。

5.4.1 收集系统高质量的信息和数据

在开发仿真模型时，分析人员应该利用所有现存的信息，包括如下几方面。

与领域专家交谈

仿真模型不是由与世隔绝工作的分析人员开发出来的抽象的模型；实际上，建模人员必须与非常熟悉系统的人紧密合作。永远不会有某个人或者文档有建模所需要的所有信息。因此，分析人员必须有丰富的资源，以得到一组完整和准确的信息。必须小心鉴别每个子系统真正的主题专家并且避免获得的数据有偏(见例 5.19)。即使仿真研究还没有做，把所有的系统信息集合到一个地方的过程本身通常也是很有价值的。注意由于系统的描述在仿真研究期间可能发生变化，建模者可能必须与某些主题专家保持经常的交流。

例 5.14 对于一个制造系统，建模者应该从源头获取信息，如机器操作人员、制造业和工业工程师、维护人员、调度人员、管理人员、供应商，以及蓝图。 ◀

例 5.15 对于一个通信网络，相关人员可能包括最终用户、网络设计者、技术专家(例如，交换机和卫星的专家)、系统管理员、应用程序架构师、维护人员、管理者和运营商等。 ◀

对系统的观测

如果与所关注的系统类似的系统已存在，那么应该从这个系统获得数据，以用于建立模型。这些数据可能可以从历史记录中获得，或者可能必须在一次研究期间收集。由于提供数据的人可能不是仿真建模人员，遵循下面两条原则是十分重要的。

- 建模者必须确保向提供数据的人精确地说明了数据需求(类型、格式、数量、采集数据的条件、为什么需要，等等)。
- 建模者必须理解产生数据的过程，而不是仅仅把观察值视为抽象的数据。

下面是关于数据的五个潜在的困难。

- 数据并不代表人们真正想要建模的东西。

例 5.16 由于军队行为特性的差异以及缺乏战场烟雾，一次军事战场测试过程中收集的数据可能不能代表实际对抗条件(另见习题 5.1)。 ◀

- 数据的类型或格式不合适。

例 5.17 在制造系统建模中，随机性的最大来源通常是机器的随机停机。实际上，我们想要的是失效时间(用实际的机器忙期表示)和机器的维修时间的数据。有时机器故障停机的数据是可以获得的，但是这些数据通常格式不恰当。例如，失效时间可能是基于墙钟时间并且包含机器空闲或下班时间。 ◀

- 数据可能包括测量、记录或者舍入误差。

例 5.18 军机部件的维修时间通常舍入到最接近的天，这使得它不可能拟合连续概率分布(见第 6 章)。 ◀

- 数据可能由于私利而“有偏”。

例 5.19 自动化工厂的维护部门报告某些机器的可靠性大于实际的，以便使自己面子上好看。 ◀

- 数据的单位可能不一致。

例 5.20 美国运输司令部通过空运、陆运和海运运送军方货物。在建立仿真模型时有时会出现混乱，因为美国空军和美国陆军采用短吨(2 000 磅)而美国海军采用长吨(2 200 磅)。 ◀

现有的理论

例如，如果对一个服务系统如银行建模，顾客的到达速率在一段时间内是固定不变的，但理论告诉我们，顾客的到达时间间隔非常类似于独立同分布的指数型随机变量；换句话说，顾客按照泊松过程到达(见第 6.12.1 小节和例 6.4)。

类似的仿真研究的相关结果

如果要建立一个军事地面遭遇战的仿真模型(像过去多次做过的那样)，那么，如果可能的话，应该寻找并利用类似的研究结果。

建模者的经验和直觉

通常有必要利用人们的经验和直觉来假设复杂系统的某些部件是如何运作的，特别是，当系统并不以某种形式存在时更应如此。人们希望这些假设能够在随后的仿真研究中得到证实。

5.4.2　与管理者定期沟通

我们现在讨论本章中最重要的思想之一，采用这种思想将大大增加完成的模型被应用于决策过程的可能性。建模者与管理者在整个仿真研究过程中保持定期沟通是非常重要的。这样做有以下好处。

- 仿真研究刚开始时，可能并没有一个如何解决问题的明确想法。因此随着研究的推进和问题的性质逐渐清晰，这些信息应该传达给管理者，他可能会重新规划研究的目标。显然，错误问题的最伟大的模型也是无效的。
- 保持管理者对研究的兴趣并参与其中。
- 管理者关于系统的知识对模型的实际有效性是有帮助的。
- 由于管理者理解并接受模型的假设，则模型更加可信。事实上，让管理者(和其他重要人员)在关键模型假设上“签字”是非常可取的。这使得管理者相信，“当然，这是一个好模型，因为我协助开发了它”。

5.4.3　维持一份书面的假设文档，并执行一次结构化走查

仿真模型含有无效的假设或者关键性疏漏的一个重要原因是沟通上的失误。所有模型概念、假设、算法和数据的文档汇总到一个书面的假设文档可以大大减轻这个问题，从而提高模型的可信性(在美国国防部，假设文件作为概念模型更为人所知)。然而，决定合适的假设文档内容是说起来容易做起来难的任务，它取决于建模者的洞察力、建模原理的知识(例如，运筹学、概率与统计等等)，以及对类似类型的系统建模的经验。假设文档不是对系统如何工作的“精确”描述，而是相对于模型在处理具体问题时是如何工作的描述。确切地说，假设文档是仿真分析人员对关注的系统应该如何进行建模的想象力的体现。

假设文档的撰写应该使得分析人员、主题专家和接受过技术培训的管理者等类似人员可读，且应该包含以下内容：

- 概述中要讨论总体项目目标、仿真研究要解决的具体问题，以及用于评估的性能度量。
- 如果适当的话，给出一个流程图或者系统布局图。
- 以符号格式详细描述每个子系统以及这些子系统是如何相互作用的(就像本页上这样，符号格式使得假设文档在结构化遍历时易于审阅，这在下文中介绍)。
- 做了哪些假设简化以及为什么要这样做。记住仿真模型是实际的简化或抽象。
- 仿真模型的局限性。
- 数据集例如样本均值以及直方图的汇总。详细的统计分析或者其他技术材料或许应该放在报告的附录中——记住假设文档对技术管理者来说应该具有可读性。
- 重要的或者有争议的信息的来源。

假设文档应该包含足够多的细节，它是创建仿真计算机程序的“蓝图”。关于假设文

档(概念模型)的其他信息可以在国防建模与仿真办公室(2011)、Pace(2003)和 Robinson (2008a，b)中找到。

如前面所讨论的那样，仿真建模者需要从许多不同的人那里收集系统信息。加之，这些人通常都非常忙于处理在他们的组织中发生的日常问题，往往导致他们对仿真建模者提出的问题给出的东西不如他们专心致志。其结果，就存在仿真建模者得不到系统的完整和正确的描述的危险。解决这个潜在问题的一个方法是在拜会主题专家或者管理者之前进行一次假设文档的结构化走查。使用投影设备，仿真建模者逐项遍历假设文档，在会议室的所有人都认可某项是正确的且详细程度适当之后才能推进到下一项。结构化走查使仿真模型的有效性和可信性都可以得到提高。

最理想的结构化走查会议应该是在一个较远的地方召开(例如宾馆的会议室)，这样人们能够把全部注意力集中在会议上。而且，会议应该在编程开始之前召开，以免大的问题未涵盖在会议中。应该在会前把假设文档发给参会人员并征询他们的意见。然而，我们并不认为这能够代替结构化走查本身，因为人们可能没有时间或者动力来自己仔细审阅文档。此外，在实际会议上产生的交互是非常有价值的[在美国国防部内部，对假设文档(概念模型)的结构化走查有时称为概念模型确认]。项目组所有关键人员都参加结构化走查以及他们都能起到积极的作用是非常必要的。

在结构化走查中恐怕许多模型假设会发现是错误的或者是缺失的。因此，在假设文档中发现的任何错误或遗漏应该在编程开始前加以纠正。

我们现在给出结构化走查的两个例子，第一个非常成功，另一个产生了非常令人惊奇的但是很有用的结果。

例 5.21 我们在对一家美国财富杂志 500 强制造公司(见第 13.6 节)的仿真研究中，执行过结构化走查。有 9 个人参加了会议，包括 2 位建模人员和 7 位来自客户组织的人员。客户人员包括机器操作者领班、3 位不同类型的工程师、2 位来自调度部门的人员以及 1 位管理者。假设文档有 19 页长，包括大约 160 个暂定模型假设。这 160 个假设中每一个都提出并讨论，整个过程持续了 5 个半小时。该过程发现了若干错误的假设并被纠正，有几条新的假设添加进去，一些细层次问题被解决。此外，在会议的最后，9 个人全部感觉他们有一个有效的模型！换句话说，他们获得了模型的所有权。 ◀

例 5.22 在一个运输系统的结构化走查中，我们的合作主办方给出的假设很大一部分被到场的主题专家发现是错误的(由于主办方和主题专家的办公室地理上距离很远，主题专家没能参加项目的启动会)。结果，不同人员分配负责收集系统不同部分的信息。收集来的信息用于更新假设文档，第二次走查成功地完成了。该经验说明了所有关键项目人员出席启动会是的非常重要的。 ◀

一些人认为通常情况下需要假设文档和正式的综述。然而，基于我们谈到的数千个仿真实践者，我们相信在所有模型中 75%的模型都没有充分的文档。

5.4.4 采用定量技术确认模型组件

仿真分析人员应该尽可能采用定量技术检测整个模型各个组件的有效性。我们现在给出一些可用于这个目的的技术的一些例子。

如果用一个理论概率分布来拟合一组观察数据，则这种表示方法的充分性可以通过在第 6 章讨论的图表和拟合优良度检验来评定。

如第 5.4.1 小节中所述，重要的是在建立模型时使用适当的数据，但是，精心组织这些数据也同样重要。举例来说，如果几组数据来自对“相同”随机现象的观察，则合并这些数据的正确性可以通过对总体的均匀性进行克鲁斯凯-沃利斯(Kruskal-Wallis)检验来评定(见第 6.13 节)。如果数据组是均匀的，则可以将他们合并且合并的数据组可以用于仿真模型的某一目的。

例 5.23 对于在第13.6节中案例研究中描述的制造系统来说，可以收集由同一个供应商生产的两台“完全相同的”机器的故障时间与维修时间，然而克鲁斯凯-沃利斯检验表明，这两个分布事实上对两台机器是不同的。因此，在仿真模型中每台机器给出了各自的故障时间和维修时间分布。

确定哪些模型因子对期望性能度量有显著影响的重要技术是灵敏度分析。如果某一因子看起来是重要的，那么它需要小心地对它建模。下面是能够用灵敏度分析研究的因子的例子：

- 参数值(见例5.24)；
- 分布的选择；
- 实体通过被仿真的系统(见例5.25)；
- 子系统的详细程度(见例5.26)；
- 对收集数据来说哪些数据是至关重要的(用一个系统的“粗糙”模型)。

例 5.24 在一个新系统的仿真研究中，假定一个参数的值与主题专家交谈的结果估计是0.75。确定该参数的重要性可以通过用0.75来运行仿真，以及再用0.7和0.8来运行仿真。如果这三次仿真运行产生几乎相同的结果，则输出对参数在0.70到0.80区间内的选择是不灵敏的。否则，就需要对这个参数给出更好的说明。◀

例 5.25 我们建立了一个块糖生产线的仿真模型。最初，我们采用单个糖块作为基本实体通过模型，但是这导致了过多的计算机执行时间。进行灵敏度测试发现，对于期望的性能度量(每班生产的箱数)来说，采用四分之一箱糖块(150块)实际上产生同样的仿真结果，同时显著减少了运行时间。◀

例 5.26 我们开发了一个PC制造企业的装配和测试区的仿真模型。后来企业管理者决定他们希望在自己的计算机上运行该模型，但是模型需要的内存太大。结果我们被迫大大简化了装配区的模型来节省计算机内存(仿真研究的主要焦点是测试区所需要的容量)。我们运行简化后的仿真模型后(测试区的模型没有改变)发现期望的性能度量，日吞吐量，与原模型相比只有2%的差别。因此，装配区的大量细节是不必要的。注意，虽然如此，简化模型并不适合研究如何提高装配区的效率。另一方面，在这种情况下，也许不必对测试区建模。◀

当人们进行灵敏度分析时，采用公共随机数方法(见第11.2节)来控制仿真中的随机性是很重要的。否则，改变一个因子的影响可能被不经意发生的其他改变(如来自某输入分布的不同随机值)混淆了。

如果人们试图确定仿真输出对改变两个或多个感兴趣的因子的灵敏度，那么，一次改变一个因子而将其他因子设定为某些任意值，一般情况下，这是不正确的。比较正确的方法是使用统计实验设计，这部分内容将在第12章中讨论。每个因子的影响可以正规地估计出来，而且如果因子的数目不是太大，因子之间相互作用也能检测出来。

5.4.5 确认整个仿真模型的输出

仿真模型有效性最权威的检验是确定仿真输出数据非常接近从实际(建议的)系统期望得到的输出数据。这称为结果确认，在这节中，我们讨论能进行结果确认的几种方法。

与已有系统比较

如果有一个系统类似于建议的系统，那么开发该现有系统的仿真模型并将仿真模型的输出数据与该现有系统本身的输出数据进行比较。如果两组数据非常“接近”，那么现有系统的模型就可认为是“有效”的(模型所需要的准确性取决于其预期的用途和管理者的效用函数)。修改模型使其代表建议的系统。现有系统和建议系统之间的共同性越大，我们对建议系统的模型的信心就越强。对确认建议系统的模型来说，不存在绝对权威的方法。如果有的话也许从一开始就不需要仿真模型了。如果上述比较是成功的，那么另一个好处是提供了应用仿真的可信性(见例5.27)。模型和系统输出数据的比较可以使用数学统计的方法来做，例如样本均值，样本方差和样本相关函数等。另外，可以采用图表(见例5.30)进行评

价，如直方图、分布函数、盒子图，以及蜘蛛网络图(在微软 Excel 中称为雷达图)。

例 5.27 我们为一个造纸公司总部进行过仿真研究。这家公司的某制造厂目前有两台某一类型的机器，当地管理部门想购置第三台机器。研究的目标是看看是否真的需要增加这台机器。为了确认我们的模型，我们首先仿真现有两台机器的系统。两台机器的模型和系统的生产量相差在 0.4%和 1.1%之间，机器利用率相差 1.7%和 11%(相对较大的误差 11%是由于第二台机器的操作者没有遵守公司的策略造成的)。采用“被确认的”仿真模型，我们仿真了三台机器的系统，发现增加的机器不是必需的。基于可信的仿真结果，负责整个公司生产的副总裁拒绝了工厂添置新机器的要求，避免了 140 万美元的投资。◀

例 5.28 一个美国空军测试部门使用后勤综合模型(LCOM)进行轰炸机机翼仿真研究。研究的最终目标是评价各种建议的后勤策略对轰炸机可用性的影响，也就是轰炸机能够执行飞行任务的时间比例。有机翼在 9 个月里实际操作的数据可用，既包括各种飞机部件的故障数据，也包括机翼可用性。为了确认模型，空军首先用现有后勤策略仿真 9 个月。模型的可用性与历史上的系统可用性的差别小于 3%，这为该模型的有效性提供了有力的证据。◀

例 5.29 一个电信交换机大制造商提出了一个原型交换机用于实验性人工话务流(例如，指数到达间隔时间)。然后将这个交换机的仿真模型也用于相同的话务流，并比较了模型和系统的性能度量。各自的度量的接近度使模型开发者相信了模型有效性。◀

例 5.30 美国陆军正在开发一种假想的新型地对地导弹。8 个原型导弹按照相同场景(和一组环境条件)进行了现场测试，并记录了他们在 Oxy 坐标系中的弹着点。建立了导弹系统的仿真模型，模型对相同场景使用不同随机数进行了 15 次独立重复运行，计算出相应的弹着点。实测和仿真的导弹命中点(以英尺为单位)如图 5.2 所示。从图看到仿真导弹不如实测导弹精确，但是也许有希望进一步证实。我们下面用勾股定理计算每个实测导弹和每个仿真的导弹的脱靶距离 d，公式如下

$$d = \sqrt{x^2 + y^2}$$

得到的脱靶距离(以英尺为单位)在表 5.4 中给出，从中可以看到，仿真的导弹的平均脱靶距离比实测导弹的平均脱靶距离大 14.7%。脱靶距离的蜘蛛网络图如图 5.3 所示。

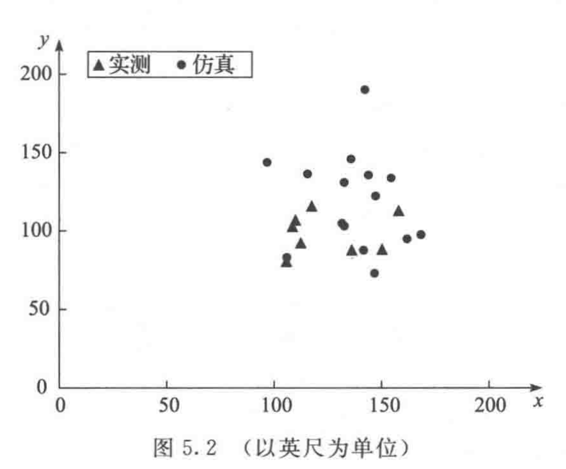


图 5.2　(以英尺为单位)

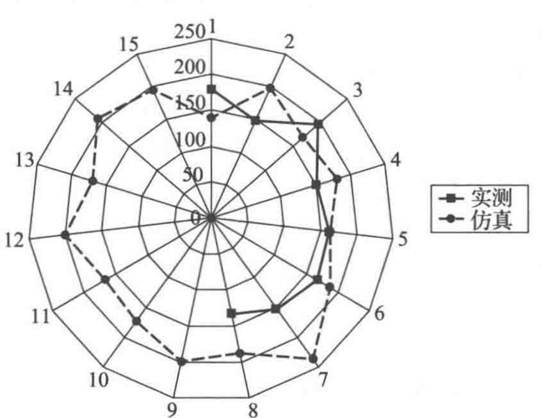


图 5.3　实测与仿真的导弹的脱靶距离的雷达图

表 5.4　测试和仿真导弹的脱靶距离 d(英尺)

导弹编号	实测脱靶距离	仿真脱靶距离
1	174.45	134.60
2	146.09	194.73
3	194.72	168.14

（续）

导弹编号	实测脱靶距离	仿真脱靶距离
4	149.84	178.82
5	161.93	163.78
6	165.52	186.39
7	153.62	237.20
8	133.46	187.73
9	—	197.90
10	—	173.55
11	—	166.64
12	—	199.10
13	—	168.17
14	—	204.32
15	—	191.48
样本均值	159.95	183.50
样本方差	355.75	545.71

数字 50，100，…，250 是脱靶距离，数字 1，2，…，15 是导弹编号。从图 5.4 可以清晰地看出，仿真脱靶距离一般比测试脱靶距离大。总之，基于样本均值和两张图，以脱靶距离为判定准则，模型没能有效代表原型导弹。不过，我们在第 5.6.2 小节中再次讨论这个例子。◀

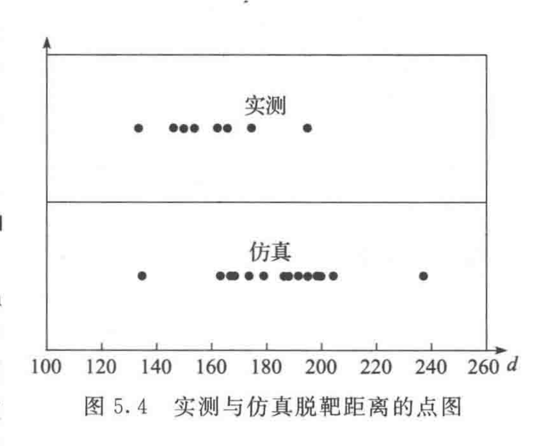


图 5.4　实测与仿真脱靶距离的点图

除了统计程序外，人们还可以利用图灵(Turing)测试[见 Turing(1950)，Schruben(1980)和 Carson(1986)]进行模型和系统的输出数据比较。要求对系统有丰富知识的人(例如工程师或管理人员)来检查一组或多组系统数据，以及一组或多组模型数据，并不让他们知道哪组是系统数据，哪组是模型数据。每组数据应该采用完全相同的格式用单独一页纸提交。如果这些主题专家可以区分系统数据和模型数据，那么他们如何做到这一点的解释可以用于改进模型。

例 5.31 Schruben(1980)报告了一个在汽车部件厂的仿真研究中使用了图灵测试。来自工厂和仿真的数据按照公式定额形式整理，并在有三个管理者、三个工业工程师和两个工厂工人的会议上审核。这些人无法就哪些数据是真实的，哪些是仿真得出的达成一致，这使得仿真模型立即被接受。◀

例 5.32 一个图灵测试的动画版本应用于确认高速公路上微观车流的仿真模型。仿真的交通流动画与从实际高速公路上收集到的数据制作的动画同时显示在大屏幕上。高速公路数据是通过飞机上安装的摄像机收集的。◀

到目前为止，我们讨论了确认仿真模型是与过去或者现在系统输出数据相关的；然而，可能更决定性的模型测试是确定其预测未来系统行为的能力。由于模型通常随时间变化而变化并为多个应用所使用(尤其是美国国防部的遗留模型)，往往有这种预测确认的机会。例如，如果模型用于决定应该建立哪个版本的系统，那么，当系统已经建立并经过了足够的时间来收集输出数据，这些数据可以与模型的预测相比较。如果合乎情理的一致，

我们就增加了对模型的“有效性”的信心。另一方面，两组数据的不一致之处应用于修改模型。不管模型的以往预测的准确性如何，在每个新的应用前应该仔细检查模型，因为目的的变化或者时间的推移都可能使得现存模型的某些方面失效。这又一次说明了好的模型文档的必要性。

假设我们比较现存系统和该系统仿真模型的数据并发现了显著差异。如果这些差异或者其他信息客观地表明了如何改进模型，那么应该进行这些改变并重新运行仿真。如果新的仿真输出数据与系统输出数据相比很接近，那么模型可以认为是“有效的”。

假设系统和模型的输出数据间不存在显著差异，但仍对模型进行了某些没有理由的改变(如“稍稍调整”某些参数)，再将结果输出数据与系统输出数据进行比较。这个过程，称为对模型的校准，一直持续到两个数据集基本吻合为止。不过，我们必须探究这个过程是否产生了系统的一个有效模型，或者，用通常的说法，该模型是否仅仅代表这组特定的输入数据。为了回答这个问题(实际上，为了确认模型)，人们可以使用完全独立的一组系统输入和输出数据。用第二组输入数据驱动被校准的模型(用类似与第 5.6.1 小节中描述的方法)并将得到的模型输出数据与第二组系统输出数据进行对比。这种使用一组数据用于校准，另一组独立数据用于确认的思想在经济学和生物学领域是相当普遍的。特别是，Crown Zellerbach 公司在开发一个树木生长的仿真模型时使用了这种方法。这里系统数据可以从美国林务局获得。

例 5.33 为了更加清晰的诠释模型校准的概念，考虑表 5.5 所示的 x、y 数据，该数据同样用于图 5.5 中。利用最小二乘法分别用线性回归模型和六阶多项式回归模型对该数据进行拟合，其结果同样显示在图 5.5 中。注意，线性回归模型并没有完全覆盖 7 个点，但是校准多项式模型完全覆盖了所有点(R^2 的值分别为 0.97 和 1)。因此，从这个意义上讲，多项式模型对数据的拟合效果更好。

如果我们利用这两个模型预测 $x'=1.5$ 所对应的 y 的值，则线性模型和(过拟合)多项式模型预测的值分别为 5.46 和 2.86，而前一个预测值似乎要更为合理。◀

表 5.5 校准示例数据

x	y
1.1	5.5
2.3	6.1
3.2	10.1
4.5	11.7
5.9	15.1
6.4	16.8
7.1	19.4

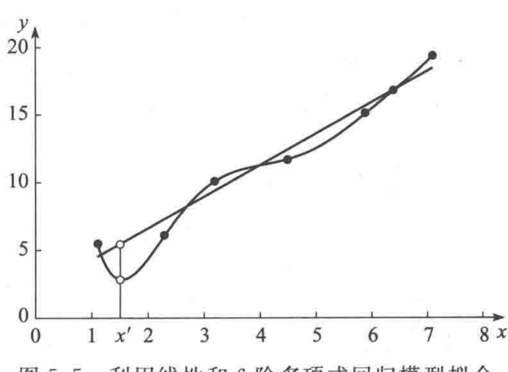


图 5.5 利用线性和 6 阶多项式回归模型拟合数据，预测 $x'=1.5$ 时对应的 y 值

与专家意见相比较

无论是否有一个实际系统，领域专家应该审查仿真结果的合理性(在执行这项工作时必须小心，因为如果有人确切知道期望的输出数据，那模型也许就没有存在的必要了)。如果仿真结果与所感知的系统行为是一致的，那么这个模型称为具有外观有效性的。

例 5.34 上述思想被很好的应用于美国空军人力和人事系统仿真模型的开发中(设计这个模型的目的是为空军政策分析人员提供各种建议的人事政策的影响的全系统评价)。模型在基本的人事政策下运行，其结果展示给空军分析人员和决策制定者，随后他们鉴别出模型和所感知的系统行为的不一致之处。这些信息用于改进模型，在几个额外的评估和改进之后，获得了一个看起来与现在的空军政策非常接近的模型。这项工作不仅增强了模型

的有效性而且提高了模型的可信性。◀

与另一个模型相比较

假设对同一个系统为“相似”目的开发了另一个模型，且这个模型认为是系统一个“有效的”代表。那么，当前感兴趣的模型数据统计或图表可以非正式地与另一个模型的可比的数据统计和图表相对比。或者，可用在第10.2节中讨论的置信区间程序进行两个模型更正式地比较。应该牢记仅仅因为两个模型产生类似的结果并不一定意味着任何一个模型是有效的，因为两个模型都可能包含相似的错误。

例5.35 国防后勤局建立一个名为性能和需求影响仿真的新的仿真模型来替代现有模型。两个模型的目标之一是决定对每个存储数量何时订货以及订多少。为了确认旧模型，将1996财政年度的所有订单的总金额与同一时期实际系统的总金额相比较。由于金额差异小于3%，旧模型的有效性有相当的可信性。为了确认新模型，两个模型都用于预测1998财政年度的所有订单总金额，结果差异小于6%。因此我们有理由相信新模型的有效性。◀

在例5.35中，对仿真分析人员来说，为了确认的目的，也使用如某类库存量的总金额这样比较小的综合水平可能会是个好主意(某些种类的正误差可能会抵消其他种类的负误差)。同样地，也许有兴趣比较两个模型在1996年发生的所有订单的总金额。

5.4.6　动画

动画是能够发现无效的模型假设和提高仿真模型可信性的有效途径。

例5.36 开发一个糖果包装系统的仿真模型。一个最近晋升的业务管理者，他不熟悉仿真模型，在第一次观看了该系统的动画后宣称“这是我的系统!”——模型立即获得了可信性。◀

5.5　管理者在仿真过程中的作用

所关注的系统的管理者必须对仿真有一个基本的理解，且意识到一个成功的仿真研究需要他/她投入时间和资源。下面是管理者的责任。

- 问题目标形式化。
- 指导全体人员为仿真建模人员提供信息和数据并参加结构化走查。
- 与仿真建模人员定期沟通。
- 应用仿真结果来辅助决策制定过程。

仿真研究在某个时期需要组织的技术人员。如果研究是在机构内部完成，那么一些公司员工可能需要付出数月的全部时间。这些人通常有其他工作如负责制造系统的日常操作。即使由专家来进行这项研究，公司员工仍然必须参与建模过程并且可能需要他们来收集数据。

5.6　比较实际观测值和仿真输出数据的统计程序

本节我们给出可能用于进行模型和系统的输出数据比较的统计程序(见第5.4.5小节)。

假设 R_1，R_2，…，R_k 是来自实际系统的观测值，M_1，M_2，…，M_k 是相应仿真模型的输出数据(见例5.37)。我们想要采取某种方法比较两组数据来确定模型是否能准确地代表实际系统。想到的第一个方法是采用经典统计检验(t、曼-惠特尼(Mann-Whitney)、双样本 χ^2、双样本科尔莫戈罗夫-斯米尔诺夫(Kolmogorov-Smirnov)检验等)之一来确定两组数据的基本分布是否的确可以视为是相同的[为了更好讨论这些检验，假定它们是独立同分布数据，见Breiman(1973)和Conover(1999)]。然而，几乎所有实际系统和仿真的输出过程是非平稳的(连续观测值的分布随时间变化而改变)和自相关的(过程中观测值是彼此相关的)，因此这些检验都不能直接应用。进一步，我们要问，当用构造差值的置信区间的方法进行比较时，假设检验还是合适的统计方法吗？由于模型只是实际系统的近似

值，原假设认为系统和模型是“相同的”，这显然是不成立的。我们认为，更为有用的问题是，系统和模型之间的差异是否足够显著，以至于影响从模型得出的结论。在第 5.6.1 小节到第 5.6.3 小节中我们分别讨论针对这种比较问题的检测法、置信区间法和时间序列方法。最后，在第 5.6.4 小节中讨论基于回归分析和步步为营法的两种其他方法。

5.6.1 检测法

似乎被大多数试图进行上述比较的仿真人员所使用的方法是计算一个或多个由实际观测数据统计量与相应的模型输出数据的统计量，然后未用正规的统计程序就比较两组统计量(见例 5.27 和例 5.28)。用于该目的的统计量的例子是样本均值、样本方差(对自相关数据使用样本方差的危险性的讨论见第 4.4 节)，以及样本相关性函数(图表的比较可能也是非常有用的，见例 5.30)。这种检测方法的困难之处在于每个统计量本质上是来自某个基础总体的样本大小为 1，例 5.37 的图表说明了这一点，这使得这种方法难以处理来自实际系统和仿真模型两者的观测值的内在随机性。

例 5.37 为了说明使用检测法的危险性，假设关注的实际系统是一个 $M/M/1$ 队列，其 $\rho=0.6$，相应的仿真模型是 $M/M/1$ 队列，$\rho=0.5$；两者的到达速率都为 1。假设关注的输出过程是 D_1，D_2，…(其中 D_i 是队列中第 i 个顾客的延误时间)，且令系统的

$$X=\frac{\sum_{i=1}^{200} D_i}{200}$$

模型的

$$Y=\frac{\sum_{i=1}^{200} D_i}{200}$$

因此，系统的观测值的个数 k 以及模型观测值的个数 l 都等于 200。我们将试图确定模型代表系统的良好程度，办法是通过将估计值 $\mu_Y=E(Y)=0.49$[模型前 200 个顾客的期望平均延误时间，关于如何计算 $E(Y)$ 的讨论见 Heathcote 和 Winer(1969)]与估计值 $\mu_X=E(X)=0.87$ 进行比较。表 5.6 给出了三次独立仿真实验的结果，每个对应于检测方法的一个可能的应用。对于每次实验，$\hat{\mu}_X$ 和 $\hat{\mu}_Y$ 表示系统和模型的 200 个延误时间的样本均值，$\hat{\mu}_X-\hat{\mu}_Y$ 是 $\mu_X-\mu_Y=0.38$ 的一个估计，这是我们真正试图要估计的。注意随着实验 $\hat{\mu}_X-\hat{\mu}_Y$ 变化很大，还看到实验 2 的 $\hat{\mu}_X-\hat{\mu}_Y=-0.01$，这恐怕会导致人们认为该模型很好地表达了系统。然而，我们认为对于估计实际系统期望平均延误时间来说该模型实际上是个很差的表达，因为 μ_Y 几乎比 μ_X 小 44%。

表 5.6 检测法的三次实验结果

实验	$\hat{\mu}_X$	$\hat{\mu}_Y$	$\hat{\mu}_X-\hat{\mu}_Y$
1	0.90	0.70	0.20
2	0.70	0.71	−0.01
3	1.08	0.35	0.73

◀

由于使用上述基本检测方法的内在危险性，我们现在介绍一个更好的比较系统和模型输出数据的方法，如果系统数据足够完整且格式正确的话，特别地，建议在比较系统和模型时用历史的系统输入数据(例如，实际观测的到达间隔时间和服务时间)，而不是用输入概率分布的样本值来“驱动”模型，随后比较模型和系统的输出，如图 5.6 所示(系统输出对应于历史的系统输入数据)。这样，系统和模型用的是输入随机变量完全相同的观测

值，这应该得出一个在统计上更精确的比较。我们称这种方法为相关检测法，因为其通常导致相比较的模型和系统的输出统计量正相关。这种方法相对于概率分布法来说是确认仿真模型的假设的更为确定性的方法；后者是用第 6 章的技术来确认的(注意，用历史的输入数据驱动的仿真有时称为跟踪驱动仿真)。

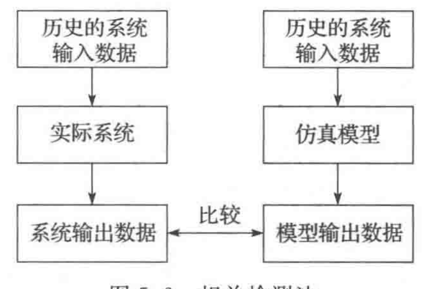


图 5.6　相关检测法

例 5.38 为了说明相关检测法的优点，假设系统是第 2.6 节中可以换队的五出纳台银行，模型是同样的银行但是不能换队(即顾客不会离开他们最开始加入的队列)。但是，假定平均服务时间现在是 4 分钟。令

X 为系统的队列中平均延误时间，

且

Y 为模型的队列中平均延误时间。

我们将试图通过比较模型的期望平均延误时间的估计值 $\mu_Y=E(Y)$ 和系统的期望平均延误时间的估计值 $\mu_X=E(X)$ 来确定模型的精确性。表 5.7 给出了 500 次独立实验的前 10 次的结果，每个对应于相关检测法一种可能的应用。其中，X_j 和 Y_j 分别是在第 j 次实验系统和模型的平均延误时间，X_j-Y_j 是我们实际想要估计的 $\mu_X-\mu_Y$ 的一个估计值(注意，X_j 和 Y_j 使用完全相同的到达间隔时间和服务时间；它们只是在是否应用换队规则上不同)。表中还给出了 Y'_j，即在第 j 次实验用独立随机数来产生到达间隔时间和服务时间的模型的平均延误时间，以及 $X_j-Y'_j$，其均值也是 $\mu_X-\mu_Y$[注意，X_j 和 Y'_j 是基于相同输入概率分布的独立的(因此是不同的)实现]。比较 Y'_j 和 X_j 大致相应于基本检测法的一个应用(在基本检测方法的一个实际应用中，输入概率分布应该不知道且必须从系统输入数据来估计)。最后，表的后两行给出了通常的每一列样本均值和样本方差，是从所有 500 次实验中计算得到的。观察该表，作为对 $\mu_X-\mu_Y$ 的估计，X_j-Y_j 比 $X_j-Y'_j$ 要好得多，因为其方差要小得多(0.08 与 4.08)。因此，相关检测法的一个特定应用 $X-Y$ 的差值，比起基本检测法的特定应用 $X-Y'$ 的差值，看来与 $\mu_X-\mu_Y$ 要接近得多。

表 5.7　相关检测法和基本检测法的 500 次实验前 10 次结果，以及对全部 500 次的求和

实验 j	X_j	Y_j	Y'_j	X_j-Y_j	$X_j-Y'_j$
1	3.06	3.81	2.62	−0.75	0.44
2	2.79	3.37	2.05	−0.58	0.74
3	2.21	2.61	4.56	−0.40	−2.35
4	2.54	3.59	1.86	−1.05	0.68
5	9.27	11.02	2.41	−1.75	6.86
6	3.09	3.75	1.85	−0.66	1.24
7	2.50	2.84	1.13	−0.34	1.37
8	0.31	0.71	3.12	−0.40	−2.81
9	3.17	3.94	5.09	−0.77	−1.92
10	0.98	1.18	1.25	−0.20	−0.27
全部 500 次实验的样本均值	2.10	2.85	2.70	−0.75	−0.60
全部 500 次实验的样本方差	2.02	2.28	2.12	0.08	4.08

◀

我们现在更清楚的解释为什么 $\mathrm{var}(X-Y)$ 比 $\mathrm{var}(X-Y')$ 要小。特别地，若 A 和 B 是随机变量，则可以证明(见习题 4.13)

$$\mathrm{var}(A-B)=\mathrm{var}(A)+\mathrm{var}(B)-2\mathrm{cov}(A,B)$$

在基本检测法的情况下，$A=X$，$B=Y'$，$\mathrm{cov}(X, Y')=0$(估计值是 0.03，见习题 4.29)，则

$$\mathrm{var}(X-Y')=\mathrm{var}(X)+\mathrm{var}(Y')$$

对相关检测法来说，$A=X$，$B=Y$，$\widehat{\mathrm{cov}}(X,Y)=2.11[\widehat{\mathrm{cov}}(X,Y)=0.99]$，则

$$\begin{aligned}\mathrm{var}(X-Y)&=\mathrm{var}(X)+\mathrm{var}(Y)-2\mathrm{cov}(X,Y)\\&=\mathrm{var}(X)+\mathrm{var}(Y')-2\mathrm{cov}(X,Y)\\&<\mathrm{var}(X-Y')\end{aligned}$$

假设真实协方差的符号与其估计值相同。

在相同统计条件下比较模型和对应系统的方法类似于在仿真中应用称为公共随机数的方差减少技术(见第 11.2 节)和在统计试验设计中应用分块技术。然而，应该说明的是，我们不建议为生产运行而使用历史的系统输入数据来驱动模型(见第 6.1 节)。

例 5.39 相关检测法用于帮助确认 Brown & Williamson 烟草公司的一个香烟制造过程的仿真模型[细节见 Carson(1986)和 Carson 等人(1981)]。制造系统基本上由一台制烟机、一个香烟的存储器(缓存)、一个包装机和一个装箱机组成。制烟机和包装机常常受到诸如卷烟纸撕裂等产品引发的故障影响。研究的主要目标是确定存储器的最优容量，这有助于减少上述故障的影响。

观察了现存系统超过 4 小时，收集了制烟机和包装机的故障时间和维修时间数据，以及总香烟产量。这些故障时间和维修时间用于驱动仿真模型进行一次 4 小时的仿真运行，并观察模型的总香烟产量。模型产量与实际产量相差仅 1% 的事实有助于增强对模型有效性的信心。◀

例 5.40 针对例 5.32 中的高速公路仿真，应用相关检测法来比较仿真模型和系统的平均行驶时间。模型用汽车进入时间、速度、行车线等来驱动，这些来自对实际系统的观察。◀

总的来说，我们认为检测法可以对某些仿真研究的仿真模型适用性给出有价值的评价(特别是当可以用相关检测法时)。事实上，对大多数研究来说，由于实际系统操作可获取数据总量的严重限制，这将是唯一可行的统计方法。虽然如此，像例 5.37 表明的那样，在解释这种方法(特别是最初的版本)的结果时必须特别小心。

5.6.2 基于独立数据的置信区间法

我们现在介绍一种在系统和模型都能收集到大量数据组情况下比较模型与相应系统的更可靠的方法。这可能是这样一种情况，例如，当感兴趣的系统部署在实验室里(见例 5.29)。但是，这种方法不适合大多数军事和制造的情况，由于实际数据很少。

按照终止型仿真的做法(见第 9.3 节、第 9.4 节和第 10 章)，假设我们从系统收集了 m 组独立数据，从模型收集了 n 组独立数据。令 X_j 为定义在第 j 组系统数据上的随机变量，令 Y_j 为定义在第 j 组模型数据上的相同的随机变量(对例 5.37 来说，X_j 是第 j 次实验的系统的队列中平均延误时间)。X_j 系列是独立同分布随机变量(假设 m 组系统数据是均匀的)，其均值 $\mu_X=E(X_j)$，Y_j 系列是独立同分布随机变量(假设 n 组模型数据是独立重复运行产生的)，其均值 $\mu_Y=E(Y_j)$。我们将试图通过构建一个 $\zeta=\mu_X-\mu_Y$ 的置信区间来比较模型和系统。基于以下原因，我们相信构建一个 ζ 的置信区间比测试原假设的 H_o：$\mu_X=\mu_Y$ 要好。

- 由于模型只是系统的一个近似，H_0 对几乎所有的情况明显为假。
- 置信区间比相应的假设检验提供了更多的信息。如果假设检验表明 $\mu_X\neq\mu_Y$，则置信区间不仅可以给出这个信息而且指出 μ_X 与 μ_Y 差异的大小。构建 ζ 的置信区间是通过置信区间来比较两个系统的问题的特殊情形，正如 10.2 节中讨论的那样。因此，我们既可以用双 t 法，也可以用韦尔奇(Welch)法来构建 ζ 的置信区间(在第 10.2 节的符号中，$n_1=m$，$n_2=n$，$X_{1j}=X_j$，$X_{2j}=Y_j$)。双 t 法要求 $m=n$，但允许 X_j 与 Y_j 是相关的，此时应该是以使用相关检测法为基础的情形(见第 5.6.1 小节)。韦尔奇方法可用于 $m\geqslant 2$ 且 $n\geqslant 2$ 的任意值，但要求 X_j 与 Y_j 独立。

Runciman、Vagenas 和 Corkal(1997)采用双 t 方法来帮助确认一个地下采矿的模型。对他们的模型来说，X_j 是第 j 个月($j=1,2,3$)每班采矿吨数的平均值。

假设我们采用双 t 法或者韦尔奇法构建了一个 $100(1-\alpha)\%$ 的 ζ 置信区间，我们令 $l(\alpha)$ 和 $u(\alpha)$ 分别为相应的置信区间下界与上界。若 $0\notin[l(\alpha),u(\alpha)]$，则 μ_X 和 μ_Y 之间的观测差，即 $\overline{X}(m)-\overline{Y}(n)$，称为是在 α 水平上统计显著的。这等价于拒绝原假设 H_0：$\mu_X=\mu_Y$ 即支持在相同 α 水平上的双边替代假设 H_1：$\mu_X\neq\mu_Y$。若 $0\in[l(\alpha),u(\alpha)]$，则任何 μ_X 和 μ_Y 之间观测差在 α 水平上不是统计显著的，可能会被解释为采样波动。即使 μ_X 和 μ_Y 之间观测差是统计显著的，对实际目标来说，也不一定意味着模型是系统的“无效的”代表。例如，若 $\zeta=1$ 但 $\mu_X=1\,000$ 且 $\mu_Y=999$，则无论我们是否发现了显著性，模型和系统之间存在的差异恐怕没有实际重要性。若模型和系统之间差异的“幅度”足够大，以至于从模型得出的关于这个系统的任何推断都是无效的，我们才会说模型和系统之间差异是实际显著的。显然，决定模型和系统之间的差异是否是实际显著的是很主观的，取决于诸如模型的目的、模型使用者的效用函数这些因素。

如果 ζ 置信区间的长度没有小到足以确定实际显著性，就必须获得额外的 X_j 或者 Y_j(或者两者都需要)。然而，请注意，对韦尔奇法来说，仅仅增加 X_j 或者 Y_j 不可能使置信区间任意小。因此，如果系统数据的组数 m 不能增加，仅仅越来越多地重复运行模型可能无法确定实际显著性。

例 5.41 假设 X_j 和 Y_j 如例 5.38 中定义，我们想要采用双 t 法构建 $\zeta=\mu_X-\mu_Y$ 的 90%置信区间来判定模型(不换队)是否是系统(可换队)的准确抽象。令 $W_j=X_j-Y_j$ 且 $m=n=10$，我们从表 5.6 的前 10 行可得到如下结果：

$$\overline{W}(10)=\overline{X}(10)-\overline{Y}(10)=2.99-3.68$$
$$=-0.69\quad(\zeta\text{ 的点估计})$$
$$\widehat{\mathrm{var}}[\overline{W}(10)]=\frac{\sum_{j=1}^{10}[W_j-\overline{W}(10)]^2}{10\times 9}$$
$$=0.02$$

ζ 的 90%置信区间为：

$$\overline{W}(10)\pm t_{9,0.95}\sqrt{\widehat{\mathrm{var}}(\overline{W}(10)}=-0.69\pm 0.26$$

或[−0.95，−0.43]。由于区间不包含 0，μ_X 和 μ_Y 之间的观测差是统计显著的。仍然需要确定这样一个差值是否是实际显著的。◀

例 5.42 考虑例 5.30 中导弹系统和相应的仿真模型。令

X_j 为第 j 个测试导弹的脱靶距离，$j=1,2,\cdots,8$；

Y_j 为第 j 个仿真导弹的脱靶距离，$j=1,2,\cdots,15$。

由于 $m=8\neq15=n$，我们采用韦尔奇法(见 10.2.2 小节)来构建一个 $\zeta=\mu_X-\mu_Y$ 的 95%置信区间。我们得到

$$\overline{X}(8)=159.95,\quad \overline{Y}(15)=183.50$$
$$S_X^2(8)=355.75,\quad S_Y^2(15)=545.71$$
$$\hat{f}=17.34\quad(\text{估计的自由度})$$

且一个 $\zeta=\mu_X-\mu_Y$ 的 95%置信区间为：

$$\overline{X}(8)-\overline{Y}(15)\pm t_{\hat{f},0.975}\sqrt{\frac{S_X^2(8)}{8}+\frac{S_Y^2(15)}{15}}=-23.55\pm 18.97$$

或[−42.52，−4.58]。由于区间不包含 0，测试导弹的平均脱靶距离和仿真导弹的平均脱靶距离的观测差是统计显著的。这个差值的实际显著性必须由相关主题专家来确定。◀

上述重复运行方法的两个困难是，它可能需要大量的数据(每组输出数据仅产生一个

“观察值”)且它无法提供关于两个输出过程自相关结构的信息(如果感兴趣的话)。

5.6.3 时间序列方法

在本节中我们简短地讨论比较模型输出数据和系统输出数据的三种时间序列方法[时间序列是一个随机过程的有限实现。例如排队模型(见例 5.37)或系统的延误时间 D_1，D_2，…，D_{200} 构成一个时间序列]。这些方法对每类输出数据仅仅需要一组，且可以产生两个输出过程自相关结构的信息。因此，上述提到的重复运行方法的两个难点在这里都不存在。然而，却存在其他重大的困难。

谱分析法[见 Fishman 与 Kiviat(1967)以及 Naylor(1971，第 247 页)]首先对每个输出过程计算样本的频谱，即估计的自协方差函数的傅里叶(Fourier)余弦变换，然后利用已有理论来构建两个频谱对数差的置信区间。这个置信区间可以潜在地用于评估两个自相关函数的相似度。该方法的两个缺点是，它需要输出过程是协方差平稳的(这一假设通常在实际中并不满足)，以及应用该方法需要高水平的数学头脑。它也难以将这类置信区间与仿真模型的确认关联起来。

频谱分析是一种非参数的方法，其对于在时间序列中观察值的分布不做假设。Hsu 和 Hunter(1977)提出了一种替代的方法，这种方法包括对每组输出数据拟合一个参数时间序列模型[见 Box、Jenkins 和 Reinsel(2008)]，以及随后应用假设检验来观察两个模型看起来是否相同。如上所述，我们认为假设检验的方法不如基于置信区间的方法理想。

Chen 和 Sargent(1987)给出了一种基于 Schruben 的标准时间序列方法(见第 9.5.3 小节)，以构建系统的稳态均值和相应仿真模型的稳态均值之差的置信区间。与第 5.6.2 小节中的方法相比，该方法一个有吸引力的特点是，只需要系统的一组输出数据和模型的一组输出数据。然而，这种方法需要两组输出数据是独立的且满足某些其他假设。

5.6.4 其他方法

Kleijnen、Bettonvil 和 Van Groenendall(1998)开发了一种基于回归分析的假设检验方法来检验组合原假设，在跟踪驱动仿真模型的情况下(见第 5.6.1 小节)，组合原假设是模型均值等于系统均值以及模型方差等于系统方差。他们的检验假定可以从系统获取 n 个(正态分布的)独立同分布观察值和从模型获取 n 个(正态分布的)独立同分布观察值，其中 $n \geqslant 3$。因此，该方法必须应用于与第 5.6.2 小节相同的情况中。他们通过进行 $M/M/1$ 队列的实验来评估该检验的统计特性(即 I 类错误的概率与幂)。

Kleijnen、Cheng 和 Bettonvil(2000，2001)，开发了一种基于步步为营法[例子见 Efron 和 Tibshirani(1993)]的自由分布的假设检验来检验原假设，在跟踪驱动的仿真模型的情况下，原假设是模型均值等于系统均值。他们的检验假定可以从系统得到 n 个独立同分布观察值，从模型中得到 sn 个独立同分布观察值，其中 $n \geqslant 3$，s 是一个由用户选择的正整数(例如，10)。因此，该方法必须应用于与第 5.6.2 小节相同的情况中。他们通过进行 $M/M/1$ 队列和其他排队系统的实验来评估该检验的统计特性。

再说一次，我们认为用置信区间的方法来确认仿真模型比用假设检验要好。

习题

5.1 如第 5.4.1 小节所述，必须注意对系统所收集的数据一定是实际想要建模的代表。有关研究将包括观察装配线上工人的效率，以便构建仿真模型，讨论该潜在问题。

5.2 讨论为什么确认一个计算机系统的模型比确认一个军事对抗模型要容易。假设关注的计算机系统类似于一个现存的系统。

5.3 如果应用第 5.6.2 小节中的置信区间法构建一个 $\zeta = \mu_X - \mu_Y$ 的置信区间，表 5.8 所示的哪些结果是可能的？

表 5.8

	统计显著	实际显著
(a)	是	是
(b)	是	No
(c)	是	?
(d)	否	是
(e)	否	否
(f)	否	?

5.4 已知下列数据，应用韦尔奇法，$m=5$ 与 $n=10$，构建一个 $\zeta=\mu_X-\mu_Y$ 的 90%置信区间。
X_j：0.92，0.91，0.57，0.86，0.90
Y_j：0.28，0.32，0.48，0.49，0.70，0.51，0.39，0.28，0.45，0.57
这个置信区间是否是统计显著的？

5.5 假设你正在仿真一个具有指数到达间隔时间的单服务台排队系统(见第 1.4 节)，并打算进行灵敏度分析，以确定使用伽马与对数正态(见第 6.2.2 小节)服务时间的效果。讨论你打算如何使用公共随机数方法(见第 11.2 节)来使得该分析统计上更精确。你的方法与相关检测法之间有什么关系？

5.6 若将表 5.6 中的 Y_j 替换为 Y'_j，重复例 5.40 中的分析。就两个置信区间效力进行评述。

5.7 假设为一个由缓冲区(队列)隔离的数量很多的机器串联组成的制造系统建立仿真模型。由于模型的计算机运行时间极长，决定将模型分为两个子模型。运行第一个子模型，并将每个零件的离去时间(以及任何其他需要的属性)写入文件中。第二个子模型的执行由存储在文件中的信息驱动。讨论这种建模方法的局限性。

5.8 为表 5.4 中的 8 个测试脱靶距离和 15 个仿真脱靶距离构建经验分布函数(定义见第 6.2.4 小节)，并在同一张图中绘制两个函数。基于这张图，测试的或者仿真的脱靶距离是否变小了？

5.9 为表 5.4 中的 8 个测试脱靶距离和 15 个仿真脱靶距离构建盒图(参见第 6.4.3 小节)。基于这个盒图，哪种脱靶距离的均值(集中趋势)更大？哪种脱靶距离的方差(偏离程度)更大？

第6章

输入概率分布的选择

6.1 引言

为使用随机输入如到达间隔时间或者需求量进行仿真，需要指定其概率分布。例如，第1.4.3小节的单服务台排队系统中，到达间隔时间是均值为1min的独立同分布指数随机变量；1.5节库存仿真的需求量指定为1、2、3、4件，对应的概率为$\frac{1}{6}$、$\frac{1}{3}$、$\frac{1}{3}$、$\frac{1}{6}$。然后，仿真模型的输入随机变量服从特定的分布，根据这些分布生成随机数，仿真随时间向前推进。第7章和第8章讨论根据不同的分布生成随机数的方法。本章讨论的是如何确定这些输入概率分布。

几乎所有实际系统都包含一个或多个随机因素源，如表6.1所示。此外，图6.1～图6.4中给出四组实际仿真项目的数据的直方图。图6.1所示对应的是汽车生产的890个机器加工时间(以分为单位)。可以看出，直方图有一个较长的右尾(正偏)，最小值近似25min。图6.2中，我们给出219个来到自助银行的到达间隔时间的直方图(以分为单位，见例6.4)。图6.3显示了856个装船时间的直方图(以天为单位，见例6.17)。最后，在图6.4中，我们给出用于制造面巾纸或卫生纸的1 000大卷纸的码数的直方图。在这种情况下，直方图有一个左长尾(负偏)。

表6.1 普通仿真应用的随机因素源

系统类型	随机因素源
制造	加工时间、机器故障时间、维修时间
国防领域	导弹或飞机的到达时间以及有效载荷、交战的结果、军火脱靶距离
通信	报文的到达间隔时间、报文类型、报文长度
运输	船舶装载时间、顾客到地铁的到达间隔时间

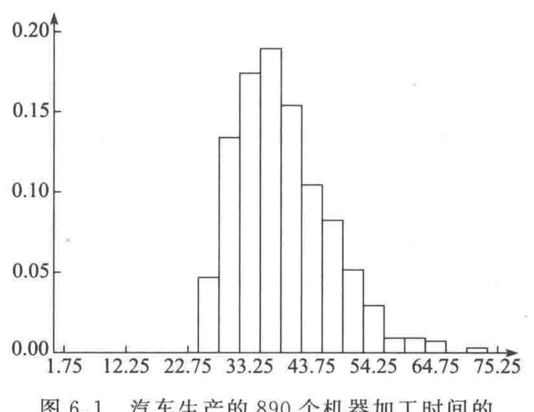


图6.1 汽车生产的890个机器加工时间的直方图

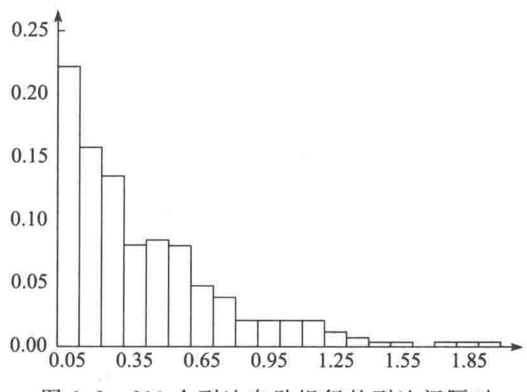


图6.2 219个到达自助银行的到达间隔时间的直方图

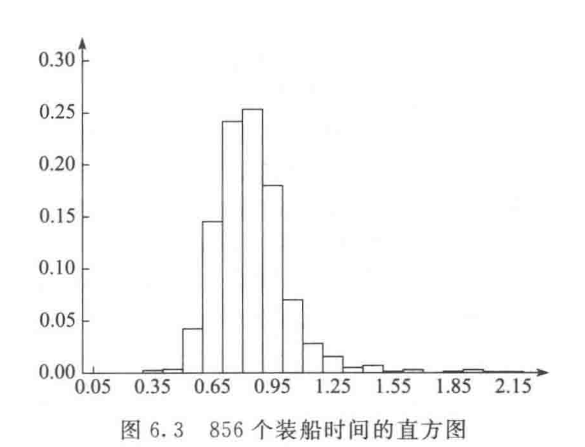


图 6.3　856 个装船时间的直方图

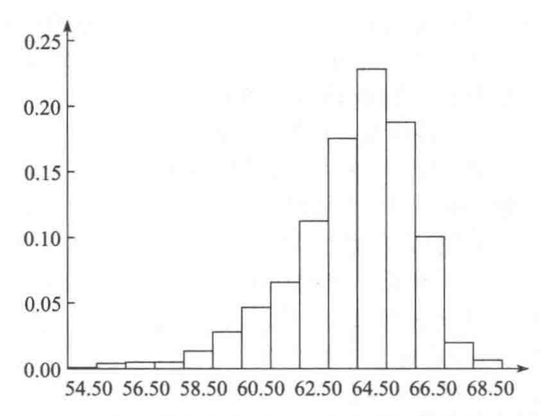


图 6.4　生活用品厂的 1 000 大卷纸码数的直方图

注意，这四个直方图没有一个像正态分布那样具有对称形状，尽管事实上很多仿真工作者和仿真书籍广泛应用正态输入分布。

从图 4.7 可以看出，在仿真模型中一般有必要使用概率分布来表示系统中的每个随机因素(而不仅仅是通过均值)。下面的例子表明，未选择“正确”分布还会影响模型的结果的精度，有时还十分严重。

例 6.1　单服务台排队系统(例如，工厂中的一台机器)具有均值为 1min 的指数到达间隔时间。假设从系统得到的 98 个服务时间可用，但其基本分布未知。使用第 6.5 节中的方法，我们为所观测的服务时间数据“拟合”出“最好的”指数分布、伽马分布、韦布尔(Weibull)分布、对数正态分布，以及正态分布(关于这些分布的讨论参见第 6.2 节，在指数分布的情况下，我们选择 β 值使得到的分布最接近“类似的”可用数据)。然后我们使用 5 个拟合分布的每一个(对正态分布，如果服务时间是负的，则再产生一次)，进行排队系统 100 次独立仿真运行(即每次运行使用不同的随机数，如在第 7.2 节中讨论的那样)。500 次仿真运行的每一次都连续运行直到收集了 10 000 个在队列中的延误时间为止。这些仿真结果数据的汇总在表 6.2 中给出。注意该表第 2 列给出了每个服务时间分布(见习题 6.27)的1 000 000个延误时间的均值。正如我们将在第 6.7 节中看到的那样，韦布尔分布实际上提供了服务时间最好的模型。因此，实际系统的平均延误时间应当接近 4.36min。另一方面，正态分布以及对数正态分布的平均延误时间分别为 6.04min 和 7.19min，对应的模型输出误差为 39%、65%。对数正态分布的结果特别出人意料，因为它具有与韦布尔分布相同一般形状(即右偏)。然而，原来对数正态分布有一个“粗”右尾部，使得更大的服务时间以及延误时间发生。表中第 4 列的“尾部概率”相对差别更加明显。显然，概率分布的选择对仿真输出有很大的影响，潜在地，对采用仿真结果进行决策的质量也就有很大的影响。◀

表 6.2　五个服务时间分布的仿真结果(合适处单位为分)

服务时间分布	在队平均延误时间	在队平均人数	延误≥15 的比例
指数分布	6.71	6.78	0.064
伽马分布	4.54	4.60	0.019
韦布尔分布	4.36	4.41	0.013
对数正态分布	7.19	7.30	0.078
正态分布	6.04	6.13	0.045

如果有可能收集到相关的输入随机变量数据，这些数据可用于以下的方法之一，以确定一个分布(按要求的增序)。

(1) 在仿真中直接使用数据值本身。例如，如果用数据代表服务时间，那么仿真中一

且需要服务时间就使用一个数据值。这有时叫做跟踪驱动仿真。

(2) 以某种方式使用数据值本身来定义一个经验分布函数(见第 6.2.4 小节)。如果这些数据代表服务时间，当仿真中需要服务时间时，我们就可以从该分布中采样。

(3) 使用统计推理的标准方法对数据“拟合”一个理论分布形式(见例 6.1)，例如指数分布或泊松分布，并进行假设检验以确定拟合的优良度。如果具有一定参数值的某个特定的理论分布是服务时间数据的好模型，那么当仿真中需要服务时间时，我们就可以从该分布中采样。

方法(1)的两个缺点是仿真只能重新产生历史上已经出现的数据，以及这些数据不足以进行所有所要求的仿真运行。方法(2)避免了这些缺点，因为至少对于连续数据来说，观察数据点的最大值和最小值之间的任何数值都可以生成(见第 8.3.16 小节)。因此，方法(2)一般好于方法(1)。然而，方法(1)也有它的用处。例如，假设想要比较一个分销中心建议的物料储运系统与现存系统。对每个进料订单来说都有到达时间、所要求产品的清单，以及每个产品的数量。对某段时间(例如一个月)的订单流建模，使用方法(2)或者方法(3)，即使不是不可能，也会是十分困难的。因此，在这种情况下，现有的和建议的系统往往使用历史订单流进行仿真。方法(1)也被推荐用于模型确认，此时，将现有系统的模型的输出与系统本身的输出进行比较(参见第 5.6.1 小节关于相关检测法的讨论)。

如果能找到某个理论分布相当好地拟合了观察数据分布(方法(3))，较之使用经验分布(方法(2))，方法(3)一般会更好一些，原因如下。

- 经验分布函数可能存在某些“不规则性”，特别是，如果只有数量很小的可用数据值时更是如此。另一方面，理论分布对数据进行了平滑，且可以提供整个基本分布的信息。
- 如果按照通常的方法使用经验分布(见第 6.2.4 小节)，就不可能在仿真中生成观察数据范围之外的数据(见第 8.3.16 小节)，这是很遗憾的，因为被仿真的系统很多性能度量都严重地依赖于“极端”事件发生的概率，例如，生成非常大的服务时间。然而，使用拟合的理论分布就可以生成观察数据值范围之外的数据。
- 在某些场合下，可能有不可克服的物理原因，使用某种理论分布形式作为特定输入随机变量的模型(见第 6.12.1 小节)。即使当我们有幸足够拥有这类信息，但为使用这类特定的分布提供实验支持，使用观察数据也是一种好想法。
- 理论分布是表达一组数据值的简洁方法。相反，如果连续分布有 n 个数据值可用，那么，在仿真软件包中，为表示经验分布，必须在计算机中输入并存储 $2n$ 个数据(例如，数据和对应的累积概率)。因此，如果该数据集很大，使用经验分布会很麻烦。
- 理论分布易于更改。例如，假设一组到达间隔时间可以比较好地建模成均值为 1 分钟的指数分布。如果我们想确定将到达速率增加 10% 对所仿真的系统的影响，那么，我们需要做的只是将指数分布的均值改为 0.909 即可。

肯定有这样的情况，即没有理论分布能够提供对观察数据的恰当的拟合，包括以下情况：

- 数据由两个或者两个以上的异构总体混合而成(见第 6.4.2 小节关于机器维修时间的讨论)。
- 完成某项任务的时间明显地被四舍五入(效果上使数据离散化)，样本值的差异不足以用任何连续理论分布来很好地代表。

在没有合适的理论分布可用的情况下，我们建议使用经验分布。理论分布(如对数正态分布)的另一个可能的缺陷是可能生成任意大的数，虽然其概率很小。因此，若已知随机变量取值不可能大于 b，那么有可能就会在 b 处截断拟合的理论分布，例如，已知银行的服务时间极不可能超过 15min。

本章的其余部分讨论与输入分布选择有关的各个主题。第 6.2 节讨论理论分布如何参数化，给出常用的连续及离散分布相关情况的概要介绍，并讨论如何确定经验分布。在第 6.3 节我们提供用于确定数据是否是某些基本分布的独立观察的一些方法，这些方法在本章很多统计程序中都需要。在第 6.4 节至第 6.6 节中我们讨论在对基本观察数据确定一

个理论分布中的三个基本活动。在第 6.7 节讨论 ExpertFit 分布拟合软件和一个综合性的例子。在第 6.8 节我们说明在某些情况下，某些理论分布，例如伽马分布、韦布尔分布，以及对数正态分布，如何能从 0 处位移使得它们更好地拟合我们的观测数据；我们还讨论了截断分布。我们在第 6.9 节讨论贝塞尔(Bézier)分布，它是基于观察数据确定分布的第四种方法。在第 6.10 节我们介绍当观测数据可用时如何确定并估计多元概率分布。在第 6.11 节我们介绍当没有数据可用时，指定输入分布的几种可能的方法。用于描述顾客到达系统方式的几个有用的概率模型在第 6.12 节中给出，而在第 6.13 节我们提供了一种确定不同来源的观测数据是否异构，以及能否合并的判定方法。

本章给出的图形及拟合优良度检验是用 ExpertFit 分布拟合软件得到的。

6.2　常用的概率分布

本节的目标是讨论可用于仿真建模的各种分布并统一列表提供这些分布的相关性能[也可参见 Evans、Hastings 与 Peacock(2000)；Johnson、Kotz 与 Balakrishman(1994，1995)，以及 Johnson、Kotz 与 Kemp(1992)]。第 6.2.1 小节对连续分布定义、参数化的常用方法进行简短讨论。然后，第 6.2.2 小节以及第 6.2.3 小节介绍几个连续及离散分布。最后，第 6.2.4 小节给出了如何能直接使用数据本身以定义经验分布的建议。

6.2.1　连续分布的参数化

对于一类给定的连续分布，例如正态或伽马分布，通常有几种不同的方法来定义或参数化概率密度函数。但是，如果参数正确地进行了定义，根据它们的物理或者几何解释，可以将参数归类为三种基本类型中的一类：位置、比例或形状。

位置参数 γ 指定了分布取值范围的横坐标(x 轴)位置点；通常 γ 是分布的范围的中点(例如正态分布的均值 μ)或者是下端点(见第 6.8 节)。在后一种情况下，位置参数有时称为位移参数。随着 γ 变化，相关的分布只是向左或者向右移动，而无其他变化。而且，若随机变量 X 的位置参数为 0，那么随机变量 $Y=X+\gamma$ 的分布的位置参数为 γ。

比例参数 β 决定了分布范围取值的测量比例(或单位，标准差 σ 是标准正态分布的比例参数)。β 变化就压缩或者放大相应的分布，而不改变其基本形状。而且，若随机变量 X 的比例参数为 1，那么随机变量 $Y=\beta X$ 的分布的比例参数为 β。

形状参数 α 不同于位置参数和比例参数，它决定了相关分布一般类别内的基本形状。α 的变化一般会改变分布的性质(如歪斜)，较之位移或者比例参数，其变化是更根本性的。某些分布(如指数分布或者正态分布)都没有形状参数，而其他分布(如贝塔分布)可以有两个形状参数。

6.2.2　连续分布

表 6.3 给出了 13 种连续分布与仿真建模应用相关的信息。给出的可能应用首先是说明分布的某些(当然不是全部)用途[其他应用见 Hahn 和 Shapiro(1994)，以及 Lawless(2003)]。然后列出密度函数和分布函数(如果存在简单的封闭形式)。接着是参数的简短说明，包括其可能的取值。范围说明相关随机变量能取值的区间。还列出了均值(期望值)、方差和众数(即密度函数取最大的值)。MLE 是指参数的最大概率似然估计，在后面的第 6.5 节进行讨论。一般注解包括所指分布同其他分布的关系。给出了每个分布的密度函数图。每个分布名字的后面的符号是我们对该分布缩写。符号“～”读作“的分布为”。

注意，我们包括了一些不太熟悉的约翰逊 S_B(Johnson S_B)分布，约翰逊 S_U(Johnson S_U)分布，对数逻辑斯蒂(loglogistic)分布，V 型皮尔逊(Pearson)分布，VI 型皮尔逊(Pearson)分布，因为我们发现这些分布较之诸如伽马分布、对数正态分布以及韦布尔这些标准分布对数据集拟合得更好。

6.2.3　离散分布

表 6.4 所示的 6 个离散分布的描述方式与表 6.3 所示的连续分布描述一样。

表 6.3 连续分布

均匀分布	$U(a, b)$
可能的应用领域	用于一个量的"初始"模型，只知道该量是在 a 与 b 之间随机变化而其余则知道得很少。$U(0, 1)$分布是由所有其他分布生成随机数的基础(见第 7 章和第 8 章)
密度函数(见图 6.5)	$f(x)=\begin{cases}\dfrac{1}{b-a}, & a\leqslant x\leqslant b\\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}0, & x<a\\ \dfrac{x-a}{b-a}, & a\leqslant x\leqslant b\\ 1, & x>b\end{cases}$
参数	a，b 均为实数，且 $a<b$；a 为位置参数，$b-a$ 为比例参数
范围	$[a, b]$
均值	$\dfrac{a+b}{2}$
方差	$\left(\dfrac{b-a}{12}\right)^2$
众数	不唯一存在
MLE	$\hat{a}=\min\limits_{1\leqslant i\leqslant n} X_i$，$\hat{b}=\max\limits_{1\leqslant i\leqslant n} X_i$
注解	(1) $U(0, 1)$均匀分布是贝塔分布的一种特殊形式($\alpha_1=\alpha_2=1$) (2) 如果 $X\sim U(a, b)$，而且$[x+\Delta x]$是$[0, 1]$的子区间，$\Delta x\geqslant 0$，有 $P(X\in[x,x+\Delta x])=\int_x^{x+\Delta x}1\mathrm{d}y=(x+\Delta x)-x=\Delta x$ 这证明了其名字"均匀"的合理性

图 6.5 $U(a, b)$密度函数

指数分布	$\text{expo}(\beta)$
可能的应用领域	"顾客"以不变的速率到达系统的到达间隔时间，每台设备发生故障的时间
密度函数(见图 6.6)	$f(x)=\begin{cases}\dfrac{1}{\beta}\mathrm{e}^{-\frac{x}{\beta}}, & x\geqslant 0\\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}1-\mathrm{e}^{-\frac{x}{\beta}}, & x\geqslant 0\\ 0, & \text{其他}\end{cases}$
参数	比例参数 $\beta>0$
范围	$[0, +\infty)$
均值	β
方差	β^2
众数	0
MLE	$\hat{\beta}=\overline{X}(n)$
注解	(1) 指数分布 $\text{expo}(\beta)$是伽马分布与韦布尔分布两者的一种特殊形式(在两者的情形下，形状参数 $\alpha=1$ 并有比例参数 β) (2) 如果 X_1，X_2，…，X_m 是独立的 $\text{expo}(\beta)$随机变量，那么，$X_1+X_2+\cdots+X_m\sim\Gamma(m, \beta)$，也称作 m-Erlang(β)分布 (3) 指数分布是唯一具有无记忆性的连续分布

图 6.6 expo(1)密度函数

（续）

伽马分布	$\Gamma(\alpha, \beta)$
可能的应用领域	完成某项任务的时间，例如顾客服务时间以及机器维修时间
密度函数(见图 6.7)	$f(x)=\begin{cases}\dfrac{\beta^{-\alpha}x^{\alpha-1}e^{-\frac{x}{\beta}}}{\Gamma(\alpha)}, & x>0\\ 0, & \text{其他}\end{cases}$ 其中，$\Gamma(\alpha)$为伽马函数，定义为：且对于任意的实数 $z>0$，$\Gamma(\alpha)=\int_0^{+\infty} t^{z-1}e^{-t}dt$。伽马函数的性质有：对于任意 $z>0$，有 $\Gamma(z+1)=z\Gamma(z)$；对于任意非负整数 k，有 $\Gamma(k+1)=k!$；对于任意正整数 k，有 $\Gamma\left(k+\frac{1}{2}\right)=\sqrt{\pi}\cdot 1\cdot 3\cdot 5\cdots(2k-1)/2^k$；$\Gamma\left(\frac{1}{2}\right)=\sqrt{\pi}$
分布函数	如果 α 不是一个整数，那么没有封闭形式；如果 α 是一个正整数，那么 $F(x)=\begin{cases}1-e^{-\frac{x}{\beta}}\sum_{j=0}^{\alpha-1}\frac{(x/\beta)^j}{j!}, & x>0\\ 0, & \text{其他}\end{cases}$
参数	形状参数 $\alpha>0$，比例参数 $\beta>0$
范围	$[0, +\infty)$
均值	$\alpha\beta$
方差	$\alpha\beta^2$
众数	$\begin{cases}\beta(\alpha-1), & \alpha\geqslant 1\\ 0, & \alpha<1\end{cases}$
MLE	必须满足以下两个方程： $\ln\hat{\beta}+\Psi(\hat{\alpha})=\dfrac{\sum_{i=1}^{n}\ln X_i}{n}, \quad \hat{\alpha}\hat{\beta}=\overline{X}(n)$ 上面的方程可以数值求解[$\Psi(\hat{\alpha})=\Gamma'(\hat{\alpha})/\Gamma(\hat{\alpha})$，称为双伽马函数；$\Gamma'$表示 Γ 的导数]。另一种方法，令 $T=[\ln\overline{X}(n)-\sum_{i=1}^{n}\ln X_i/n]^{-1}$，可得到近似值$\hat{\alpha}$与$\hat{\beta}$，即利用表 6.21(见附录 6A)，可得到$\hat{\alpha}$表示为 T 的函数，并令$\hat{\beta}=\overline{X}(n)/\bar{\alpha}$[关于本方法的推导以及表 6.21 的推导，见 Choi 和 Wette(1969)]
注解	(1) expo(β)和 $\Gamma(1, \beta)$分布函数相同 (2) 对于一个正整数 m，$\Gamma(m, \beta)$分布称作 m-Erlang(β)分布 (3) k 自由度的 χ^2 分布与 $\Gamma(k/2, \beta)$分布相同 (4) 如果 X_1，X_2，…，X_m 是独立随机变量，$X_i\sim\Gamma(\alpha_i, \beta)$，那么 $X_1+X_2+\cdots+X_m\sim\Gamma(\alpha_1+\alpha_2+\cdots+\alpha_m, \beta)$ (5) 如果 X_1，X_2 是独立的随机变量，$X_i\sim\Gamma(\alpha_i, \beta)$，那么 $X_1/X_1+X_2\sim\beta(\alpha_1, \alpha_2)$ (6) $X\sim\Gamma(\alpha, \beta)$当且仅当 $Y=1/X$ 服从皮尔逊 V 型分布，且形状、比例参数分别为 α、$1/\beta$，记作 PT5(α, $1/\beta$) (7) $\lim\limits_{x\to 0}f(x)=\begin{cases}+\infty, & \alpha<1\\ \frac{1}{\beta}, & \alpha=1\\ 0, & \alpha>1\end{cases}$

图 6.7　$\Gamma(\alpha, 1)$密度函数

（续）

韦布尔分布	$W(\alpha,\ \beta)$
可能的应用领域	完成某项任务的时间，一台设备的故障时间，在缺少数据的情况下用作粗略模型(见第 6.11 节)
密度函数(见图 6.8)	$f(x)=\begin{cases}\alpha\beta^{-\alpha}x^{\alpha-1}\mathrm{e}^{-\left(\frac{x}{\beta}\right)^{\alpha}}, & x>0\\ 0, & 其他\end{cases}$
分布函数	$F(x)=\begin{cases}1-\mathrm{e}^{-\left(\frac{x}{\beta}\right)^{\alpha}}, & x>0\\ 0, & 其他\end{cases}$
参数	形状参数 $\alpha>0$，比例参数 $\beta>0$
范围	$[0,\ +\infty)$
均值	$\frac{\beta}{\alpha}\Gamma\left(\frac{1}{\alpha}\right)$
方差	$\frac{\beta^2}{\alpha}\left\{2\Gamma\left(\frac{2}{\alpha}\right)-\frac{1}{\alpha}\left[\Gamma\left(\frac{1}{\alpha}\right)\right]^2\right\}$
众数	$\begin{cases}\beta\left(\frac{\alpha-1}{\alpha}\right)^{1/\alpha}, & \alpha\geqslant 1\\ 0, & \alpha<1\end{cases}$
MLE	必须满足以下两个方程 $\frac{\sum_{i=1}^{n}X_i^{\hat{\alpha}}\ln X_i}{\sum_{i=1}^{n}X_i^{\hat{\alpha}}}-\frac{1}{\hat{\alpha}}=\frac{\sum_{i=1}^{n}\ln X_i}{n},\quad \hat{\beta}=\left(\frac{\sum_{i=1}^{n}X_i^{\hat{\alpha}}}{n}\right)^{1/\hat{\alpha}}$ 第一个方程中的$\hat{\alpha}$可以通过牛顿(Newton)法数值求解，然后第二个方程直接给出$\hat{\beta}$。牛顿迭代的一般回归步骤为： $\hat{\alpha}_{k+1}=\hat{\alpha}_k+\frac{A+1/\hat{\alpha}_k-C_k/B_k}{1/\hat{\alpha}_k^2+(B_kH_k-C_k^2)/B_k^2}$ 其中， $A_k=\frac{\sum_{i=1}^{n}\ln X_i}{n},B_k=\sum_{i=1}^{n}X_i^{\hat{\alpha}_k},C_k=\sum_{i=1}^{n}X_i^{\hat{\alpha}_k}\ln X_i$ 且 $H_k=\sum_{i=1}^{n}X_i^{\hat{\alpha}_k}\ (\ln X_i)^2$ [关于这些公式以及 α，β 的置信区间参见 Thoman，Bain 以及 Antle(1969)]作为迭代的开始，可能会用到估计： $\hat{\alpha}_0=\left\{\frac{(6/\pi^2)\left[\sum_{i=1}^{n}(\ln X_i)^2-\left(\sum_{i=1}^{n}\ln X_i\right)^2/n\right]}{n-1}\right\}^{-1/2}$ [因为 Menon(1963)以及 Thoman、Bain 与 Antle(1969)中建议过]根据上式选择$\hat{\alpha}_0$，Thoman、Bain 与 Antle(1969)报告过，平均只需要 3.5 次迭代就达到 4 位精度
注解	(1) expo(β)和 $W(1,\ \beta)$分布是相同的 (2) $X\sim W(1,\ \beta)$当且仅当 $X^{\alpha}\sim \mathrm{expo}(\beta^{\alpha})$(见习题 6.2) (3) Weibull 随机变量的(自然)对数分布称为极值(extreme-value)分布或者冈贝尔(Gumbel)分布[请参考文献：Averill M. Law & Associates(2006)，Lawless(2003)以及习题 8.1(b)]

a）

b）

图 6.8　$W(\alpha,\ 1)$密度函数

（续）

韦布尔分布	$W(\alpha,\ \beta)$	
注解	(4) $W(2,\ \beta)$也称为参数为β的瑞利(Rayleigh)分布，记作$R(\beta)$。如果Y和Z为均值0方差β^2的独立正态随机变量(参见正态分布)，那么$X=(Y^2+Z^2)^{1/2}\sim R(2^{1/2}\beta)$ (5) 随着$\alpha\to+\infty$，韦布尔分布在β处退化。因此大α的韦布尔密度函数在众数处有一个尖峰 (6) 当$\alpha>3.6$时，韦布尔分布有负偏[参见图 6.8(b)] (7) $\lim_{x\to 0} f(x)=\begin{cases}+\infty, & \alpha<1\\ \dfrac{1}{\beta}, & \alpha=1\\ 0, & \alpha>1\end{cases}$	

正态分布	$N(\mu,\ \sigma^2)$
可能的应用领域	各种类型的错误，如炸弹的弹着点；大量数值的累加和(中心极限定理)
密度函数(见图 6.9)	$f(x)=\dfrac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/(2\sigma^2)}$，对于所有实数$x$
分布函数	无封闭形式
参数	位置参数$\mu\in(-\infty,\ +\infty)$，比例参数$\sigma>0$
范围	$(-\infty,\ +\infty)$
均值	μ
方差	σ^2
众数	μ
MLE	$\hat{\mu}=\overline{X}(n)$，$\hat{\sigma}=\left[\dfrac{n-1}{n}S^2(n)\right]^{1/2}$
注解	(1) 如果两个联合分布的正态随机变量不相关，那么它们也是独立的。对于非正态分布，一般说来，这种隐含不成立 (2) 假设X_1，X_2，…，X_m的联合分布为多元正态分布，且令$\mu_i=E(X_i)$和$C_{ij}=\text{cov}(X_i,\ X_j)$。那么对于任意的实数$a$，$b_1$，$b_2$，…，$b_m$，随机变量$a+b_1X_1+b_2X_2+\cdots+b_mX_m$也服从正态分布，且均值为$\mu=a+\sum_{i=1}^{m}b_i\mu_i$且方差为 $$\sigma^2=\sum_{i=1}^{m}\sum_{j=1}^{m}b_ib_jC_{ij}$$ 注意：我们不必假定X_i是独立的。若X_i相互独立，那么 $$\sigma^2=\sum_{i=1}^{m}b_i^2\text{var}(X_i)$$ (3) $N(0,\ 1)$分布常常称为标准或单位正态分布 (4) 若X_1，X_2，…，X_k为独立标准正态分布随机变量，那么$X_1^2+X_2^2$，…，X_k^2服从自由度为k的χ^2分布，也是$\Gamma(k/2,\ 2)$分布 (5) 若$X\sim N(\mu,\ \sigma^2)$，那么e^X为对数正态分布，比例参数为e^μ，形状参数为σ，记作$LN(\mu,\ \sigma^2)$ (6) 若$X\sim N(0,\ 1)$，且Y为自由度为k的χ^2分布，且X与Y独立，那么$X/\sqrt{Y/K}$为自由度为k的t分布(有时也称为学生t分布) (7) 若使用正态分布来表示一个非负量(例如，时间)，那么其密度应该在$X=0$处截断(见第 6.8 节) (8) 随着$\sigma\to 0$，正态分布退化到μ

图 6.9　$N(0,\ 1)$密度函数

（续）

对数正态分布	$LN(\mu, \sigma^2)$
可能的应用领域	完成某项任务的时间［密度函数形状同 $\alpha>1$ 时的 $\Gamma(\alpha, \beta)$ 与 $W(\alpha, \beta)$，但是在 $x=0$ 附近有一个大的“尖峰”，十分有用］；大量数值的乘积量（依据中心极限定理）；用作缺少数据时的粗略模型（见第 6.11 节）
密度函数（见图 6.10）	$f(x)=\begin{cases}\dfrac{1}{x\sqrt{2\pi\sigma^2}}\exp\dfrac{-(\ln x-\mu)^2}{2\sigma^2}, & x>0\\ 0, & 其他\end{cases}$
分布函数	无封闭形式
参数	形状参数 $\sigma>0$，比例参数 $e^{\mu}>0$
范围	$[0, +\infty)$
均值	$e^{\mu+\sigma^2/2}$
方差	$e^{2\mu+\sigma^2}(e^{\sigma^2}-1)$
众数	$e^{\mu-\sigma^2}$
MLE	$\hat{\mu}=\dfrac{\sum_{i=1}^{n}\ln X_i}{n}$，$\hat{\sigma}=\left[\dfrac{\sum_{i=1}^{n}(\ln X_i-\hat{\mu}^2)}{n}\right]^{1/2}$
注解	（1）$X \sim LN(\mu, \sigma^2)$ 当且仅当 $\ln X \sim N(\mu, \sigma^2)$。因此，如果有数据 $X_1, X_2, \cdots, X_n$ 被认为是对数正态分布，这些数据点的对数 $\ln X_1, \ln X_2, \cdots, \ln X_n$ 可以按正态分布数据对待，以用于分布假设、参数估计，以及拟合优良度检验中 （2）随着 $\sigma \to 0$，正态分布退化到 e^{μ}，因此，σ 很小时，对数正态分布的密度函数在众数处有一个峰值 （3）无论参数值是什么，$\lim\limits_{x\to 0} f(x)=0$

图 6.10　$LN(0, \sigma^2)$ 密度函数

贝塔分布	$\beta(\alpha_1, \alpha_2)$
可能的应用领域	用于缺少数据时的粗略模型（参考第 6.11 节）；随机比例分布，例如装货中货物损坏件数的比例；完成某项任务的时间，例如在 PERT 网络中
密度函数（见图 6.11）	$f(x)=\begin{cases}\dfrac{x^{\alpha_1-1}(1-x)^{\alpha_2-1}}{B(\alpha_1, \alpha_2)}, & 0<x<1\\ 0, & 其他\end{cases}$ 其中，$B(\alpha_1, \alpha_2)$ 为贝塔函数，定义如下：对于任意的实数 $z_1>0$ 与 $z_2>0$ $B(z_1, z_2)=\int_0^1 t^{z_1-1}(1-t)^{z_2-1}\mathrm{d}t$ 贝塔函数的一些性质如下： $B(z_1, z_2)=B(z_2, z_1), B(z_1, z_2)=\dfrac{\Gamma(z_1)\Gamma(z_2)}{\Gamma(z_1+z_2)}$
分布函数	一般无封闭形式。如果 α_1 或者 α_2 是一个正整数，$F(x)$ 可以通过二项展开式获得，它是 x 的多项式，x 的指数一般为正实数，介于 0 和 $\alpha_1+\alpha_2-1$ 之间
参数	形状参数 $\sigma_1>0$ 与 $\sigma_2>0$
范围	$[0, 1]$
均值	$\dfrac{\alpha_1}{\alpha_1+\alpha_2}$
方差	$\dfrac{\alpha_1\alpha_2}{(\alpha_1+\alpha_2)^2(\alpha_1+\alpha_2+1)}$

a）

b）

图 6.11　$\beta(\alpha_1, \alpha_2)$ 密度函数

（续）

贝塔分布	$\beta(\alpha_1, \alpha_2)$
众数	$\begin{cases} \dfrac{\alpha_1-1}{\alpha_1+\alpha_2-2}, & \text{如果 } \alpha_1>1,\ \alpha_2>1 \\ 0 \text{ 和 } 1, & \text{如果 } \alpha_1<1,\ \alpha_2<1 \\ 0, & \text{如果}(\alpha_1<1,\ \alpha_2\geqslant 1)\text{或者}(\alpha_1=1,\ \alpha_2>1) \\ 1, & \text{如果}(\alpha_1\geqslant 1,\ \alpha_2<1)\text{或者}(\alpha_1>1,\ \alpha_2=1) \\ \text{不唯一存在}, & \text{如果 } \alpha_1=\alpha_2=1 \end{cases}$
MLE	必须满足下面两个方程： $\Psi(\hat{\alpha}_1)-\Psi(\hat{\alpha}_1+\hat{\alpha}_2)=\ln G_1, \Psi(\hat{\alpha}_2)-\Psi(\hat{\alpha}_1+\hat{\alpha}_2)=\ln G_2$ 其中，Ψ 为双伽马函数；$G_1=\left(\prod_{i=1}^{n}X_i\right)^{1/n}$，且 $G_2=\left[\prod_{i=1}^{n}(1-X_i)\right]^{1/n}$［参见 Gnanadesikan，Pinkham 以及 Hughes(1967)］；注意 $G_1+G_2\leqslant 1$。这些方程可数值求解［参见 Beckman 和 Tietjen(1978)］或者通过表 6.22(参见附录 6A)得到 $\hat{\alpha}_1$ 和 $\hat{\alpha}_2$ 的近似值，表 6.22 通过修改 Beckman 和 Tietjen(1978)中的方法得到特定 (G_1, G_2) 对的计算方法
注解	(1) $U(0, 1)$ 和 $\beta(1, 1)$ 分布是相同的 (2) 如果 X_1，X_2 是独立随机变量，$X_i\sim\Gamma(\alpha_i, \beta)$，那么 $X_1/(X_1+X_2)\sim\beta(\alpha_1, \alpha_2)$ (3) [0, 1]上的贝塔随机变量通过变换 $a+(b-a)X$ 可以改变比例和改变位置，以得到$[a, b]$上的有相同形状的贝塔随机变量 (4) $X\sim\beta(\alpha_1, \alpha_2)$ 当且仅当 $1-X\sim\beta(\alpha_1, \alpha_2)$。 (5) $X\sim\beta(\alpha_1, \alpha_2)$ 当且仅当 $Y=X/(1-X)$ 分布为皮尔逊Ⅵ型分布，形状参数为 α_1，α_2，而且比例因子为 1，记作 PT6$(\alpha_1, \alpha_2, 1)$。 (6) $\beta(1, 2)$ 密度函数为左三角分布，而 $\beta(2, 1)$ 为右三角分布。 (7) $\lim\limits_{x\to 0}f(x)=\begin{cases}+\infty, & \text{如果 } \alpha_1<1 \\ \alpha_2, & \text{如果 } \alpha_1=1, \\ 0, & \text{如果 } \alpha_1>1\end{cases}$ $\lim\limits_{x\to 1}f(x)=\begin{cases}\infty, & \text{如果 } \alpha_2<1 \\ \alpha_1, & \text{如果 } \alpha_2=1 \\ 0, & \text{如果 } \alpha_2>1\end{cases}$ (8) 当且仅当 $\alpha_1=\alpha_2$，密度函数是关于 $x=\frac{1}{2}$ 对称的。而且，当且仅当 $\alpha_1=\alpha_2$，均值与众数相等

c）

d）

图 6.11 （续）

皮尔逊Ⅴ型分布	PT5(α, β)
可能的应用领域	完成某些任务的时间(密度函数同对数正态形状相似，但是在靠近 $x=0$ 处有一个大的峰值)
密度函数(见图 6.12)	$f(x)=\begin{cases}\dfrac{x^{-(\alpha+1)}e^{-\beta/x}}{\beta^{-\alpha}\Gamma(\alpha)}, & x>0 \\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}1-F_G\left(\dfrac{1}{x}\right), & x>0 \\ 0, & \text{其他}\end{cases}$， 其中，$F_G(x)$ 为 $\Gamma(\alpha, 1/\beta)$ 随机变量的分布函数

图 6.12 PT5$(\alpha, 1)$密度函数

（续）

皮尔逊Ⅴ型分布	PT5(α, β)
参数	形状参数 $\alpha>0$，比例参数 $\beta>0$
范围	$[0, +\infty)$
均值	$\frac{\beta}{\alpha-1}$，对于 $\alpha>1$
方差	$\frac{\beta^2}{(\alpha-1)^2(\alpha-2)}$，对于 $\alpha>2$
众数	$\frac{\beta}{\alpha+1}$
MLE	如果有数据 X_1, X_2, …, X_n，对 $1/X_1$, $1/X_2$, …, $1/X_n$ 拟合 $\Gamma(\alpha_G, \beta_G)$可以得到最大似然估计$\hat{\alpha}_G$ 和$\hat{\beta}_G$。那么 PT5(α, β)的最大似然估计为 $\alpha_G=\hat{\alpha}_G$ 以及 $\hat{\beta}=1/\hat{\beta}_G$(参见下面的注解 1)
注解	(1) 当且仅当 $Y=1/X\sim\Gamma(\alpha, 1/\beta)$，$X\sim$PT5($\alpha$, β)。因此，皮尔逊Ⅴ型分布有时也称作反伽马分布 (2) 注意，均值和方差只是对某些形状参数值才存在

皮尔逊Ⅵ型分布	PT6(α_1, α_2, β)
可能的应用领域	完成某任务的时间
密度函数(见图 6.13)	$f(x)=\begin{cases}\frac{(x/\beta)^{\alpha_1-1}}{\beta B(\alpha_1, \alpha_2)[1+x/\beta]^{\alpha_1+\alpha_2}}, & x>0\\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}F_B\left(\frac{x}{x+\beta}\right), & x>0\\ 0, & \text{其他}\end{cases}$ 其中，$F_B(x)$为 $\beta(\alpha_1, \alpha_2)$随机变量分布函数
参数	形状参数 $\alpha_1>0$ 且 $\alpha_2>0$，比例参数 $\beta>0$
范围	$[0, +\infty)$
均值	$\frac{\beta\alpha_1}{\alpha_2-1}$，对于 $\alpha_2>1$
方差	$\frac{\beta^2\alpha_1(\alpha_1+\alpha_2-1)}{(\alpha_2-1)^2(\alpha_2-2)}$，对于 $\alpha_2>2$
众数	$\begin{cases}\frac{\beta(\alpha_1-1)}{\alpha_2+1}, & \alpha_1>1\\ 0, & \text{其他}\end{cases}$
MLE	如果有数据 X_1, X_2, …, X_n 认为是 PT6(α_1, α_2, 1)分布，那么对 $X_i/(1+X_i)$，$i=1, 2, \cdots, n$ 拟合 $\beta(\alpha_1, \alpha_2)$分布，得到最大似然估计$\hat{\alpha}_1$ 和$\hat{\alpha}_2$。那么 PT6(α_1, α_2, 1)(注意 $\beta=1$)分布的最大似然估计也是$\hat{\alpha}_1$ 和$\hat{\alpha}_2$(参见注解 1)
注解	(1) $X\sim$PT6(α_1, α_2, 1)当且仅当 $Y=X/(1+X)\sim\beta(\alpha_1, \alpha_2)$ (2) 如果 X_1 和 X_2 为独立随机变量，且$X_1\sim\Gamma(\alpha_1, \beta)$且 $X_2\sim\Gamma(\alpha_2, 1)$，那么 $Y=X_1/X_2\sim$PT6(α_1, α_2, β)(参见习题 6.3)

图 6.13　PT6(α_1, α_2, 1)密度函数

（续）

皮尔逊Ⅵ型分布	PT6(α_1, α_2, β)	
注解	(3) 注意，均值和方差只对某些形状参数 α_2 值才存在	d) 图 6.13 （续）
对数逻辑斯蒂分布	$LL(\alpha, \beta)$	
可能的应用领域	完成某任务的时间	
密度函数(见图 6.14)	$f(x)=\begin{cases}\dfrac{\alpha\,(x/\beta)^{\alpha-1}}{\beta[1+(x/\beta)^{\alpha}]^2}, & x>0\\ 0, & 其他\end{cases}$	
分布函数	$F(x)=\begin{cases}\dfrac{1}{1+(x/\beta)^{-\alpha}}, & x>0\\ 0, & 其他\end{cases}$	
参数	形状参数 $\alpha>0$，比例参数 $\beta>0$	
范围	$[0, +\infty)$	
均值	$\beta\theta\operatorname{csch}(\theta)$，对于 $\alpha>1$，其中，$\theta=\pi/\alpha$	
方差	$\beta^2\theta\{2\operatorname{csch}(2\theta)-\theta[\operatorname{csch}(\theta)]^2\}$，对于 $\alpha>2$	
众数	$\begin{cases}\beta\left(\dfrac{\alpha-1}{\alpha+1}\right)^{1/\alpha}, & \alpha_1>1\\ 0, & 其他\end{cases}$	
MLE	令 $Y_i=\ln X_i$，求解下面两个方程得到 $\hat{a}$ 和 $\hat{b}$： $$\sum_{i=1}^{n}\left[1+e^{(Y_i-\hat{a})/\hat{b}}\right]^{-1}=\frac{n}{2} \quad (6.1)$$ 和 $$\sum_{i=1}^{n}\left(\frac{Y_i-\hat{a}}{\hat{b}}\right)\frac{1-e^{(Y_i-\hat{a})/\hat{b}}}{1+e^{(Y_i-\hat{a})/\hat{b}}}=n \quad (6.2)$$ 那么对数的对数分布的 MLE 为 $\hat{\alpha}=1/\hat{a}$，$\hat{\beta}=e^{\hat{b}}$。Johnson、Kotz 和 Balakrishnan(1995，第 23 章)建议使用牛顿法求解式(6.1)和式(6.2)	图 6.14　$LL(\alpha, 1)$密度函数
注解	$X\sim LL(\alpha, 1)$当且仅当 $\ln X$ 分布是位置参数为 $\ln\beta$、比例参数为 $1/\alpha$ 的逻辑斯蒂(logistic)分布(参见习题 8.1)。因此，如果有数据 $X_1, X_2, \cdots, X_n$ 认为是对数逻辑斯蒂分布，那么数据点的对数 $\ln X_1, \ln X_2, \cdots, \ln X_n$ 视为逻辑斯蒂分布，以用于分布假设、参数估计和拟合优良度检验	

（续）

约翰逊 S_B分布	JSB(α_1，α_2，a，b)
密度函数（见图 6.15）	$f(x)=\begin{cases}\frac{\alpha_2(b-a)}{(x-a)(b-x)\sqrt{2\pi}}e^{-\frac{1}{2}\left[\alpha_1+\alpha_2\ln\frac{(x-a)}{(b-x)}\right]^2}, & a<x<b \\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}\Phi\left[\alpha_1+\alpha_2\ln\left(\frac{(x-a)}{(b-x)}\right)\right], & a<x<b \\ 0, & \text{其他}\end{cases}$ 其中，Φ为$\mu=0$，$\sigma^2=1$的正态随机变量的分布函数
参数	位置参数$a\in(-\infty,+\infty)$，比例参数为$b-a(b>a)$，形状参数$\alpha_1\in(-\infty,+\infty)$，且$\alpha_2>0$
范围	$[a, b]$
均值	各阶矩都存在，但是都特别复杂[参见 Johnson，Kotz，和 Balakrishnan(1994，第 35 页)]
众数	当$\alpha_2<\frac{1}{\sqrt{2}}$且$\lvert\alpha_1\rvert<\frac{\sqrt{1-2\alpha_2^2}}{\alpha_2}-2\alpha_2\,\text{artanh}(\sqrt{1-2\alpha_2^2})$时，密度函数具有 2 个尖峰[artanh 为双曲正切函数的反函数]；否则密度函数为单峰函数。除了范围端点外，任何众数x满足方程： $\frac{2(x-a)}{b-a}=1+\alpha_1\alpha_2+\alpha_2^2\ln\left(\frac{x-a}{b-x}\right)$
注解	(1) $X\sim \text{JSB}(\alpha_1, \alpha_2, a, b)$当且仅当： $\alpha_1+\alpha_2\ln\frac{(x-a)}{(b-x)}\sim N(0,1)$ (2) 如果$\alpha_1>0$，$\alpha_1=0$，$\alpha_1<0$，密度函数分别为左偏、对称、右偏 (3) 对于所有的α_1，α_2，都有$\lim\limits_{x\to a}f(x)=\lim\limits_{x\to b}f(x)=0$ (4) 这四个参数可以通过许多方法估计出来[例如，参见 Swain，Venkatranman，和 Wilso(1988)以及 Slifker 和 Shapiro(1980)]
约翰逊 S_U分布	JSU(α_1，α_2，γ，β)
密度函数（见图 6.15）	$f(x)=\frac{\alpha_2}{\sqrt{2\pi}\sqrt{(x-\gamma)^2+\beta^2}}$ $e^{-\frac{1}{2}\left\{\alpha_1+\alpha_2\ln\left[\frac{x-\gamma}{\beta}+\sqrt{\left(\frac{x-\gamma}{\beta}\right)^2+1}\right]\right\}^2}$，$-\infty<x<+\infty$
分布函数	$F(x)=\Phi\left\{\alpha_1+\alpha_2\ln\left[\frac{x-\gamma}{\beta}+\sqrt{\left(\frac{x-\gamma}{\beta}\right)^2+1}\right]\right\}$， $-\infty<x<+\infty$
参数	位置参数$\gamma\in(-\infty,+\infty)$，比例参数为$\beta>0$，形状参数$\alpha_1\in(-\infty,+\infty)$，$\alpha_2>0$
范围	$[-\infty, +\infty]$
均值	$\gamma-\beta\exp(1/(2\alpha_2^2))\sinh\left(\frac{\alpha_1}{\alpha_2}\right)$，其中，sinh 为双曲正弦函数
众数	除了范围端点外，众数满足的方程为：$\gamma+\beta y$，其中，y满足： $y+\alpha_1\alpha_2\sqrt{y^2+1}+\alpha_2^2\sqrt{y^2+1}\ln(y+\sqrt{y^2+1})=0$

图 6.15 JSB(α_1，α_2，0，1)密度函数

图 6.16 JSU(α_1，α_2，0，1)密度函数

（续）

约翰逊 S_U分布	JSU(α_1, α_2, γ, β)
注解	(1) $X \sim \text{JSU}(\alpha_1, \alpha_2, \gamma, \beta)$当且仅当： $\alpha_1 + \alpha_2 \ln\left[\frac{X-\gamma}{\beta} + \sqrt{\left(\frac{X-\gamma}{\beta}\right)^2 + 1}\right] \sim N(0,1)$ (2) 如果分别 $\alpha_1 > 0$，$\alpha_1 = 0$，或 $\alpha_1 < 0$，密度函数为左偏、对称、右偏 (3) 这四个参数可以通过许多方法估计出来[例如，参见 Swain，Venkatranman，和 Wilso（1988）以及 Slifker 和 Shapiro(1980)]

图 6.16 （续）

三角分布	triang(a, b, m)
可能的应用领域	用于缺少数据时的粗略模型(参见第 6.11 节)
密度函数(见图 6.17)	$f(x)=\begin{cases}\frac{2(x-a)}{(b-a)(m-a)}, & a \leqslant x \leqslant m \\ \frac{2(b-x)}{(b-a)(b-m)}, & m < x \leqslant b \\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}0, & x<a \\ \frac{(x-a)^2}{(b-a)(m-a)}, & a \leqslant x \leqslant m \\ 1-\frac{(b-x)^2}{(b-a)(b-m)}, & m < x \leqslant b \\ 1, & b<x\end{cases}$
参数	a，b，与 m 为实数，$a<m<b$。a 为位置参数，$b-a$ 为比例参数，m 为形状参数
范围	$[a, b]$
均值	$\frac{a+b+m}{3}$
方差	$\frac{a^2+b^2+m^2-ab-am-bm}{18}$
众数	m
MLE	正如第 6.11 节描述的那样，当数据缺少时我们使用三角分布作为粗略模型，因此，MLE 没有什么意义
注解	极限情形 $m \to b$ 以及 $m \to a$ 时分别叫做右三角和左三角分布，在习题 8.7 中进行了讨论。对于 $a=0$ 和 $b=1$ 的左三角和右三角都是贝塔分布的特殊情形

图 6.17　triang(a, b, m)密度函数

表 6.4 离散分布

伯努利分布	Bernoulli(p)
可能的应用领域	两种可能结果的随机发生；用来生成其他离散随机变量，如二项分布、几何分布和负二项分布
质量函数(见图 6.18)	$p(x)=\begin{cases}1-p, & x=0\\ p, & x=1\\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}0, & x<0\\ 1-p, & 0\leqslant x<1\\ 1, & 1\leqslant x\end{cases}$
参数	$p\in(0,1)$
范围	$\{0,1\}$
众数	p
方差	$p(1-p)$
众数	$\begin{cases}0, & p<\frac{1}{2}\\ 0\text{ 和 }1, & p=\frac{1}{2}\\ 1, & p>\frac{1}{2}\end{cases}$
MLE	$\hat{p}=\hat{X}(n)$
注解	(1) Bernoulli(p)随机变量 X 可以看做实验要么“成功”，要么“失败”的一种结果。如果成功的概率为 p，并且如果实验失败我们令 $X=0$，成功的话令 $X=1$，那么 $X\in$ Bernoulli(p)。这种实验，往往称为伯努利实验，提供了一种方便的方法来将几个其他离散分布同伯努利分布关联起来 (2) 如果 t 为正整数，而且 X_1，X_2，…，X_t 为独立伯努利 Bernoulli(p)随机变量，那么 $X_1+X_2+\cdots+X_t$ 为二项分布，参数分别为 t 和 p。因此，二项随机分布可以看做是在给定独立伯努利实验次数中成功的次数 (3) 假设我们进行伯努利实验的独立重复运行，每次成功的概率为 p。那么观测到第一次实验成功之前失败的次数是参数为 p 的几何分布。对于一个正整数 s，在观测到第 s 实验成功前失败的次数是具有参数为 s、p 的负二项分布 (4) 伯努利分布是二项分布的特殊情形($t=1$，且有相同的 p 值)

图 6.18 Bernoulli(p)质量函数

离散均匀分布	$DU(i,j)$
可能的应用领域	多种可能结果随机发生，每个都具有相同的发生概率；用作当一个量在整数 i 到 j 之间变化，而又不知道更多信息的时候的“初始”模型
质量函数(见图 6.19)	$p(x)=\begin{cases}\frac{1}{j-i+1}, & x\in\{i,i+1,\cdots,j\}\\ 0, & \text{其他}\end{cases}$
分布函数	$F(x)=\begin{cases}0, & x<i\\ \frac{\lfloor x\rfloor-i+1}{j-i+1}, & i\leqslant x\leqslant j,\\ 1, & j<x\end{cases}$ 其中，$\lfloor x\rfloor$表示$\leqslant x$ 的最大整数

图 6.19 $DU(i,j)$质量函数

（续）

离散均匀分布	$DU(i, j)$
参数	i，j 为整数，$i \leqslant j$；i 为位置参数，$j-i$ 为比例参数
范围	$\{i, i+1, \cdots, j\}$
均值	$\frac{i+j}{2}$
方差	$\frac{(j-i+1)^2-1}{12}$
众数	不唯一存在
MLE	$\hat{i}=\min\limits_{1\leqslant k\leqslant n} X_k$，$\hat{j}=\max\limits_{1\leqslant k\leqslant n} X_k$
注解	$DU(0, 1)$和 Bernoulli(0.5)是相同的分布
二项分布	$B(t, p)$
可能的应用领域	每次实验成功的概率为 p，t 次独立伯努利试验中成功的次数；在一批大小为 t 件的物品中“残缺”的件数；一批(如一组人群)随机大小的个体的数目；库存系统需求物品的件数
质量函数(见图 6.20)	$p(x)=\begin{cases}\binom{t}{x}p^x(1-p)^{t-x}, & x\in\{0, 1, \cdots, t\}\\ 0, & \text{其他}\end{cases}$ 其中，$\binom{t}{x}$为二项分布的系数，定义为$\binom{t}{x}=\frac{t!}{x!\,(t-x)!}$
分布函数	$F(x)=\begin{cases}0, & x<0\\ \sum\limits_{i=0}^{\lfloor x\rfloor}\binom{t}{i}p^i(1-p)^{t-i}, & 0\leqslant x\leqslant t\\ 1, & t<x\end{cases}$
参数	t 为正整数，$p\in(0, 1)$
范围	$\{0, 1, \cdots, t\}$
均值	tp
方差	$tp(1-p)$
众数	$\begin{cases}p(t+1)-1 \text{ 和 } p(t+1), & p(t+1)\text{为整数}\\ \lfloor p(t+1)\rfloor, & \text{其他}\end{cases}$
MLE	如果 t 已知，那么$\hat{p}=\overline{X}(n)/t$。如果 t 和 p 都未知，当且仅当 $\overline{X}(n)>(n-1)S^2(n)/n=V(n)$时$\hat{t}$和$\hat{p}$存在。那么可以采用下面的方法。令 $M=\max\limits_{1\leqslant i\leqslant n}$，并且对于 $k=0, 1, \cdots M$，令 f_k 为所有 $X_i\geqslant k$ 的个数。然后，可以证明，$\hat{t}$和$\hat{p}$是使如下函数最大化的 t 与 p 的值 $g(t,p)=\sum\limits_{k=1}^{M} f_k\ln(t-k+1)+nt\ln(1-p)+n\overline{X}(n)\ln\frac{p}{1-p}$ 满足约束 $t\in\{M, M+1, \cdots\}$而且 $0<p<1$。很容易看到，对于一个固定的 t 值，譬如说 t_0，使 $g(t_0, p)$最大化的 p 值为 $\overline{X}(n)/t_0$，因此，对于 $t\in\{M, M+1, \cdots, M'\}$，$\hat{t}$和$\hat{p}$为得到 $g(t, \overline{X}(n)/t)$最大值的 t 和 $\overline{X}(n)/t$ 值，其中，M' 由下式给出[参见 DeRiggi (1983)]： $M'=\left\lfloor\frac{\overline{X}(n)(M-1)}{1-[V(n)/\overline{X}(n)]}\right\rfloor$ 注意 $g(t, \overline{X}(n)/t)$为 t 的单峰函数

图 6.20 $B(t, p)$质量函数

（续）

二项分布	$B(t, p)$	
注解	(1) 如果 $Y_1, Y_2, \cdots, Y_t$ 为独立 Bernoulli(p)随机变量，那么 $Y_1+Y_2+\cdots+Y_t \in B(t, p)$ (2) 如果 $X_1, X_2, \cdots X_m$ 为独立随机变量，而且 $X_i \in B(t_i, p)$，那么 $X_1+X_2+\cdots+X_m \sim B(t_1+t_2+\cdots+t_m, p)$ (3) 当且仅当 $p=0.5$ 时 $B(t_i, p)$分布对称 (4) 当且仅当 $t-X \sim B(t, 1-p)$时 $X \sim B(t, p)$ (5) $B(1, p)$和 Bernoulli(p)分布相同	

几何分布	geom(p)
可能的应用领域	独立伯努利实验序列中第一次成功前失败的次数，每次实验的成功概率为 p；检查到第一个残缺零件前的被检零件的个数；一批随机大小的物品中物品的件数；库存系统需求的货物的件数
质量函数(见图 6.21)	$p(x)=\begin{cases} p(1-p)^x, & x \in \{0, 1, \cdots\} \\ 0, & \text{其他} \end{cases}$
分布函数	$F(x)=\begin{cases} 1-(1-p)^{\lfloor x \rfloor+1}, & x \geqslant 0 \\ 0, & \text{其他} \end{cases}$
参数	$p \in (0, 1)$
范围	$\{0, 1, \cdots\}$
均值	$\dfrac{1-p}{p}$
方差	$\dfrac{1-p}{p^2}$
众数	0
MLE	$\hat{p}=\dfrac{1}{\overline{X}(n)+1}$
注解	(1) 如果 $Y_1, Y_2, \cdots$为独立 Bernoulli(p)随机变量序列，而且 $X=\min\{i: Y_i=1\}-1$，那么 $X \sim \text{geom}(p)$ (2) 如果 $X_1, X_2, \cdots, X_m$ 为独立 geom(p)随机变量，那么 $X_1+X_2+\cdots+X_m$ 为参数为 s 和 p 的负二项分布 (3) 在唯一具有无记忆性的离散分布的意义下，几何分布是指数分布的离散模拟(参见习题 4.27) (4) geom(p)是负二项分布的特殊形式($s=1$，且 p 值相同)

图 6.21 geom(p)质量函数 (p = 0.25; p = 0.50)

负二项分布	negbin(s, p)
可能的应用领域	在独立伯努利实验序列中，每次成功的概率为 p，第 s 次成功前失败的次数；检查到第 s 个残缺零件前检查到好零件的个数；一批随机大小的物品中物品的件数；库存系统需求的货物的件数
质量函数(见图 6.22)	$p(x)=\begin{cases} \dbinom{s+x-1}{x} p^s (1-p)^x, & x \in \{0, 1, \cdots\} \\ 0, & \text{其他} \end{cases}$
分布函数	$F(x)=\begin{cases} \displaystyle\sum_{i=0}^{\lfloor x \rfloor} \binom{s+i-1}{i} p^s (1-p)^i, & x \geqslant 0 \\ 0, & \text{其他} \end{cases}$

图 6.22 negbin(s, p)质量函数 (s = 2, p = 0.5)

（续）

负二项分布	negbin(s, p)
参数	$p\in(0,1)$，s 为正整数
范围	$\{0, 1, \cdots\}$
均值	$s\dfrac{1-p}{p}$
方差	$\dfrac{s(1-p)}{p^2}$
众数	令 $y=[s(1-p)-1]/p$，那么 众数 $=\begin{cases} y 与 y+1 & y 是整数 \\ \lfloor y \rfloor & 其他 \end{cases}$
MLE	如果 s 已知，那么 $\hat{p}=\dfrac{s}{\overline{X}(n)+s}$。如果 s 和 p 都未知，那么当且仅当 $V(n)=(n-1)S^2(n)/n>\overline{X}(n)$ 时 $\hat{s}$ 和 $\hat{p}$ 存在。令 $M=\max\limits_{1\leqslant i\leqslant n} X_i$，且对 $k=0, 1, \cdots M$，令 f_k 为 $X_i\geqslant k$ 的 X_i 个数。那么能证明 $\hat{s}$ 和 $\hat{p}$ 是使如下函数最大化的 s 和 p 值： $h(s,p)=\sum\limits_{k=1}^{M} f_k\ln(s+k-1)+ns\ln p+n\overline{X}(n)\ln(1-p)$ 满足约束 $s\in\{1, 2, \cdots\}$ 和 $0<p<1$。对于固定的 s 值，譬如说 s_0，最大化 $h(s_0, p)$ 的 p 值为 $s_0/[\overline{X}(n)+s_0]$，因此我们可以检查 $h(1, 1/[\overline{X}(n)+1])$，$h(2, 2/[\overline{X}(n)+2])$，$\cdots$，选择 $\hat{s}$ 和 $\hat{p}$ 使得 $h(s, s/[\overline{X}(n)+s])$ 是最大观测值的 s 和 $s/[\overline{X}(n)+s]$。但是，由于 $h(s, s/[\overline{X}(n)+s])$ 是 s 的单峰函数[参见 Levin 和 Reeds(1977)]，结束搜索的时间是显然的
注解	(1) 如果 Y_1，Y_2，$\cdots Y_s$ 为独立 geom(p) 随机变量，那么 $Y_1+Y_2+Y_s\sim$ negbin(s, p) (2) 如果 Y_1，Y_2，$\cdots$ 独立为 Bernoulli(p) 随机变量序列，而且 $X=\min\left\{i: \sum\limits_{j=1}^{i} Y_j = s\right\}-s$，那么 $X\sim$ negbin(s, p) (3) 如果 X_1，X_2，$\cdots X_m$ 为独立随机变量且 $X_i\sim$ negbin(s_i, p)，那么 $X_1+X_2+\cdots+X_m\sim$ negbin($s_1+s_2+\cdots s_m$, p) (4) negbin(1, p) 和 geom(p) 分布相同

图 6.22 （续）

泊松分布	$P(\lambda)$
可能的应用领域	当事件以不变的速率(参见第 6.12 节)发生时，某段时间内发生事件的次数；一批随机大小的物品的件数；一个库存系统需求货物的件数
质量函数(见图 6.23)	$p(x)=\begin{cases} \dfrac{e^{-\lambda}\lambda^x}{x!}, & x\in\{0, 1, \cdots\} \\ 0, & 其他 \end{cases}$
分布函数	$F(x)=\begin{cases} 0, & x<0 \\ e^{-\lambda}\sum\limits_{i=0}^{\lfloor x\rfloor}\dfrac{\lambda^i}{i!}, & x\geqslant 0 \end{cases}$
参数	$\lambda>0$

图 6.23　$P(\lambda)$ 质量函数

（续）

泊松分布	$P(\lambda)$
范围	$\{0, 1, \cdots\}$
均值	λ
方差	λ
众数	$\begin{cases}\lambda-1 \text{ 和 } \lambda, & \lambda \text{ 是整数} \\ \lfloor \lambda \rfloor, & \text{其他}\end{cases}$
MLE	$\hat{\lambda}=\overline{X}(n)$
注解	(1) 令 Y_1，Y_2，$\cdots Y_s$ 为非负独立同分布随机变量序列，且令 $X=\max\left\{i:\sum_{j=1}^{i} Y_j \leqslant 1\right\}$。那么 Y_i 为 expo$(1/\lambda)$，当且仅当 $X \sim P(\lambda)$。此外，如果 $X'=\max\left\{i:\sum_{j=1}^{i} Y_j \leqslant \lambda\right\}$，那么 Y_i 为 expo$(1/\lambda)$，当且仅当 $X' \sim P(\lambda)$[参见第 6.12 节] (2) 如果 X_1，X_2，$\cdots$，X_m 为独立随机变量，而且 $X_i \sim P(\lambda_i)$，那么 $X_1+X_2+\cdots+X_m \sim P(\lambda_1+\lambda_2+\cdots \lambda_m)$

图 6.23 （续）

6.2.4 经验分布

在一些情况下我们可能想使用观测数据本身直接（在某种意义下）定义一个分布，这称为经验分布。仿真时由经验分布生成随机数，而不是对数据拟合一个理论分布。例如，我们只是不能够找到足够好的理论分布来拟合数据（参考第 6.4 节和第 6.6 节）就有可能发生这种情况。本节主要研究定义经验分布的方法。

对于连续随机变量，能定义的经验分布的类型依赖于我们是否具有单个原始观测 X_1，X_2，$\cdots$，X_n 的实际值，而不是只有 X_i 落到若干规定的区间的个数（后者称为分组数据，或者称为直方图形式数据）。如果可以得到原始数据，我们就可以定义一个连续分段线性分布函数 F，首先将 X_i 从小到大排序，令 $X_{(i)}$ 为第 i 个最小的 X_j，那么 $X_{(1)} \leqslant X_{(2)} \leqslant \cdots X_{(n)}$。那么 F 可以由下式给出：

$$F(x)=\begin{cases}0, & x<X_{(1)} \\ \dfrac{i-1}{n-1}+\dfrac{x-X_{(i)}}{(n-1)(X_{(i+1)}-X_{(i)})}, & X_{(i)} \leqslant x<X_{(i+1)}, \text{对于 } i=1,2,\cdots n-1 \\ 1, & X_{(n)} \leqslant x\end{cases}$$

图 6.24 给出了 $n=6$ 时的图示。注意，在 X_i 分布集中的区域，$F(x)$上升得最快，正如所期望的那样。而且，对于每个 i，$F(X_{(i)})=(i-1)/(n-1)$，（对于大的 n）它近似于小于 $X_{(i)}$ 的 X_j 的比例；这也是我们期望连续分布函数所具有的性质（定义 F 的另外一种方法

的讨论，参见习题 6.5）。然而，定义这种特定经验分布的一个显而易见的缺点是，仿真运行中由它产生的随机值不可能小于 $X_{(1)}$ 或者大于 $X_{(n)}$（参见第 8.3.16 小节）。另外，$F(x)$ 的均值也不等于 X_i 的样本均值 $\overline{X}(n)$（参见习题 6.6）。

然而，如果数据是分组的，必须采用另外一种方法，因为我们不知道单个 X_i 的值。假设 n 个 X_i 分为 k 个相邻的区间 $[a_0, a_1)$，$[a_1, a_2)$，…，$[a_{k-1}, a_k)$，那么第 j 个区间包含了 n_j 个观测值，其中 $n_1+n_2+\cdots+n_k=n$（往往 a_j 是等距的，但是不必做这样的假设）。可以定义一个合理分段线性经验分布函数 G 的方法是：首先令 $G(a_0)=0$，并且对于 $j=1, 2, \cdots, k$，令 $G(a_j)=(n_1+n_2+\cdots+n_j)/n$，之后在 a_j 间进行线性插值，定义：

$$G(x)=\begin{cases}0, & x<a_0\\ G(a_{j-1})+\dfrac{x-a_{j-1}}{a_j-a_{j-1}}[G(a_j)-G(a_{j-1})], & a_{j-1}\leqslant x<a_j, \quad j=1,2,\cdots,k\\ 1, & a_k\leqslant x\end{cases}$$

图 6.25 展示了 $k=4$ 的经验分布的这种定义。在这种情况下，$G(a_j)$ 就是 X_i 小于 a_j 的比例，在整个 x 的范围内，观测点密度比较大的地方，$G(x)$ 上升得较快。然而，从这种分布中生成的随机数仍然限于下界（为 a_0）和上界（为 a_k）范围内，参见第 8.3.16 小节。

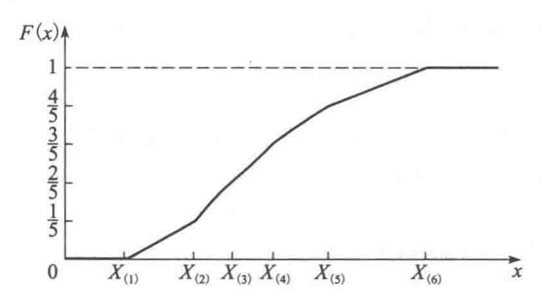


图 6.24　从原数据得到的连续、分段线性的经验分布函数

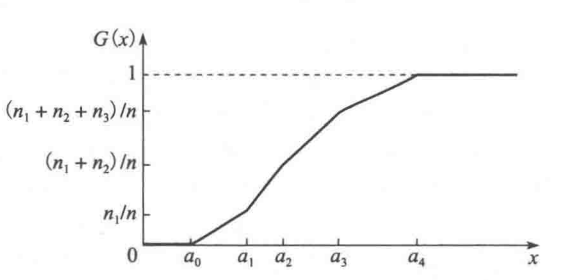


图 6.25　从分组数据得到的连续分段线性的经验分布函数

在实际中，很多连续分布是右偏的，密度函数的形状与图 6.26 所示的相似。因此，如果样本大小 n 不是很大，我们就可能即使有的话也只有很少的真实的基本分布右尾的观测值(由于这些尾部概率通常很小)。另外，上述经验分布生成的随机数不容许超过最大观测值。另一方面，非常大的生成值对仿真运行的处理具有显著的影响。例如，一个大的服务时间能够引起排队系统明显的拥塞。因此，Bratley，Fox 和 Schrage（1987，第 131-133，150-151 页）建议在经验分布右侧附加一个指数分布，这就使得容许生成大于 $X_{(n)}$ 的值。

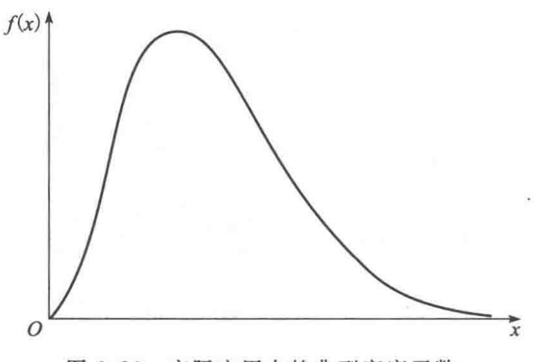


图 6.26　实际应用中的典型密度函数

对于离散的数据，倘若有原始数据 X_1，X_2，…，X_n 可用，定义一个经验分布是相当简单的。对于每个可能的值 x，经验质量函数 $p(x)$ 可以定义为 X_i 等于 x 的数量比例。对于分组离散数据，我们可以这样定义一个质量函数，即一个区间中所有可能的 x 对应的 $p(x)$ 的和等于在该区间中的 X_i 的比例。在一个区间内的 x 的可能值对应的各个 $p(x)$ 如何分布本质上是随意的。

6.3 评估样本独立性的方法

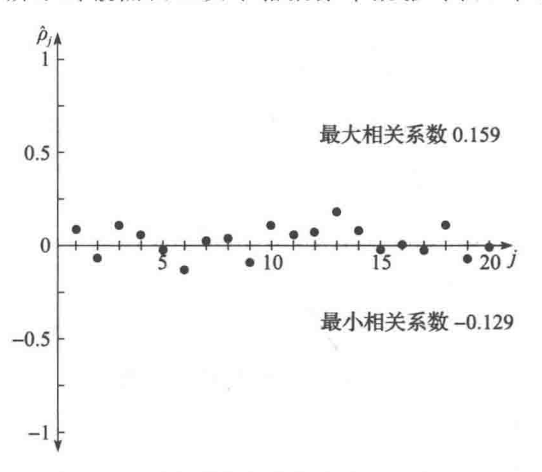
本章讨论的很多统计方法都有一个重要的假设，那就是观测值 X_1，X_2，…，X_n 是某个基本分布的独立的(或随机的)样本。例如，最大似然估计(参见第 6.5 节)和 χ^2 检验(参见第 6.6.2 小节)都假设了独立性。如果关于独立性的假设不成立，那么这些统计方法也许是无效的。然而，即使这些数据不是独立的，像直方图这样的启发式的方法仍然可以使用。

有时所搜集的观测数据在时间上是相关的。例如，假设 X_1，X_2，…表示某个城市在特定的某天中午之后每小时的温度。我们并不期望这些数据是独立的，因为时间点接近的小时气温是正相关的。作为第二个例子，考虑第 1.4 节单服务台排队系统，令 X_1，X_2，…是相继到达系统的顾客在队列中的延误时间。如果顾客到达速率接近服务时间，那么系统就会堵塞，因而 X_i 会是高度正相关的。

下面描述非正式评估数据 X_1，X_2，…(按收集的时间顺序列出)是否独立的两种图形方法。相关系数图是样本相关系数$\hat{\rho}_j$($j=1$，2，…，l，l 是正整数)的图(参见第 4.4 节)。样本相关系数$\hat{\rho}_j$ 是时间上相差 j 个观测的两个观测之间真实相关系数 ρ_j 的估计(注意$-1\leqslant\rho_j\leqslant1$)。如果观测数据 X_1，X_2，…，X_n 是独立的，那么对于 $j=1$，2，…，$n-1$，$\rho_j=0$。然而，即使 X_1，X_2，…，X_n 是独立的，$\hat{\rho}_j$ 也不一定严格地等于 0，因为$\hat{\rho}_j$ 是均值不为 0 的随机变量的观测(参见第 4.4 节)。如果$\hat{\rho}_j$ 与 0 有很大的数量差异，那么这是说明 X_i 不是独立的有力证据。

观测数据 X_1，X_2，…，X_n 的散点图是关于(X_i，X_{i+1})点对的图，其中 $i=1$，2，…，$n-1$。为简单起见假设 X_i 都非负。如果 X_i 是独立的，人们将期望点(X_i，X_{i+1})就会随机的分布在(X_i，X_{i+1})平面的第一象限内。然而，散点的性质将会依赖于 X_i 的基本分布。如果 X_i 是正相关的，那么点就会在第一象限内趋向于排成一条正斜率的直线。如果 X_i 是负相关的，那么点就会在第一象限内趋向于排成一条负斜率直线。

例 6.2 图 6.27 和图 6.28 中给出了均值为 1 的指数分布 100 个独立观测的相关系数图和散点图。注意，在图 6.27 中，样本相关系数接近 0，但是绝对值大到 0.16。图 6.28 所示的散点图证实了指数分布数据的独立性。



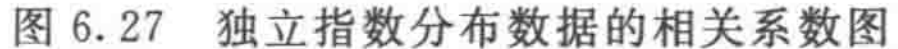


图 6.27 独立指数分布数据的相关系数图

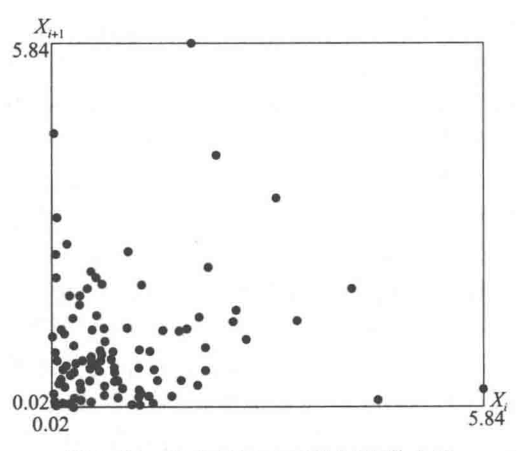


图 6.28 独立指数分布数据的散点图

例 6.3 图 6.29 和图 6.30 给出了利用率因子 $\rho=0.8$ 的 $M/M/1$ 排队系统 100 个延误时间的相关系数图和散点图(参见第 1.4.3 小节)。注意，对于小的 j 值，$\hat{\rho}_j$ 很大，散点图中的数据趋向于排成一个具有正斜率的直线。这些事实和队列中的延误时间正相关是一致的。

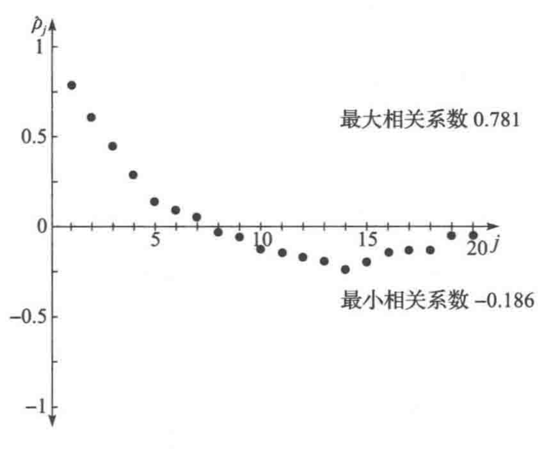


图 6.29　相关的排队数据的相关系数图

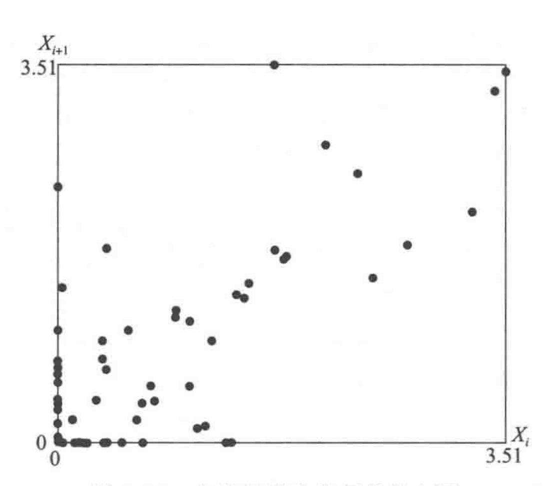


图 6.30　相关的排队数据的散点图

还有一些非参数(即对 X_i 的分布不做假设)统计检验方法，可以用来正式地检验 X_1，X_2，…，X_n 是否独立。Bartels(1982)提出了一个冯・诺伊曼比率(Von Neumann's ratio)的排序版本作为独立性的检验统计，并提供进行检验所必需的关键值。然而，一个潜在的缺陷就是该检验方法假设数据之间没有"粘连"，"粘连"的含义是指对于 $i \neq j$，$X_i = X_j$。对于离散数据来讲，这个要求一般是不满足的；对于连续数据来说，如果记录它们的精度只有很少的十进制位(参见表 6.7 中的到达间隔时间)，这个要求也不满足。Bartels 说明如果"粘连"的数目少，他的关键值可能仍然有适度的精度。

有若干运行检验的版本[例如，参见第 7.4.1 小节以及 Gibbons(1985)]，它们可以用来评估 X_i 的独立性。它们处理"粘连"遇到的困难应当比排序冯・诺伊曼检验少一些，因为运行检验只需要对于 $i=1, 2, \cdots, n-1$，$X_i \neq X_{i+1}$。另一方面，Bartels 实验性地证明了对于 X_i 独立之外其他类型的问题，排序冯・诺伊曼检验要比运行检验强大很多。

6.4　活动 Ⅰ：假设分布类别

选择一个特定输入分布的第一步就是要决定哪些类常用分布(例如，指数分布、正态分布或者泊松分布)形状上与其适合，而不考虑(暂时)这些类别的具体参数值。本节介绍用于假设分布类别的一般性方法，这些分布类别可能是仿真输入随机变量的代表。

在某些情况下，可以利用有关某个随机变量在系统中的作用的先验知识来选择一个分布进行建模，或者至少选出一些分布；这些都是基于理论来做的，完全不需要任何数据。例如，如果顾客每次一人以不变速率到达服务设施，并且在非相连时间区间内到达的顾客数量是独立的，那么假设到达间隔时间是独立同分布指数随机变量就有理论依据(参见第 6.12.1 小节)。再回忆一下，一些离散分布(二项分布、几何分布和非负二项分布)都产生于物理模型。分布的范围常常也排除了用某个分布建模。例如，服务时间不应该从正态分布中直接生成(至少在原理上如此)，因为任何正态分布的随机数有可能是负的。大批零件中残缺零件数的比例也不应假设为伽马分布，因为比例必须在 0 和 1 之间，而伽马分布随机变量没有上限。只要有先验信息就应当使用，但是也强烈地建议要用数据对假设分布进行确认。

在实际中，我们很少有足够的这类理论先验信息来选择单一的分布。由观测数据来假设分布类型的任务某种程度上不是那么定形的。本节的后续部分，将讨论各种启发式方法或指导性方法，以便于选择合适的分布类别。

6.4.1　求和统计

一些分布的特征至少部分地由其真实参数的函数来表征。表 6.5 给出了一些这样的函

数、从独立同分布数据 X_1，X_2，…，X_n 估计函数的公式(这些估计称为求和统计)、是否适用于连续(C)或者离散(D)数据的指示，以及关于它们的解释或者用法的注解(考虑到它们的效用，包括了样本最小值和最大值，即使它们也许不是分布的参数的直接函数)。关于这些函数许多内容的进一步讨论可以在第 4 章中找到。

表 6.5　有用的求和统计

函数	样本估计(求和统计)	连续(C)/离散(D)	注释
最小值、最大值	$X_{(1)}$，$X_{(n)}$	C，D	$[X_{(1)}, X_{(n)}]$是范围的粗略估计
均值 μ	$\overline{X}(n)$	C，D	中心趋向的度量
中值 $x_{0.5}$	$\hat{x}_{0.5}(n)=\begin{cases}X_{((n+1)/2)}, & 若\ n\ 为奇数\\ [X_{(n/2)}+X_{((n/2)+1)}], & 若\ n\ 为偶数\end{cases}$	C，D	另一种中心趋向的度量
方差 σ^2	$S^2(n)$	C，D	可变性的度量
方差系数 $\mathrm{cv}=\dfrac{\sqrt{\sigma^2}}{\mu}$	$\widehat{\mathrm{cv}}(n)=\dfrac{\sqrt{s^2(n)}}{\overline{X}(n)}$	C	备选的可变性的度量
莱克塞斯比率 $\tau=\dfrac{\sigma^2}{\mu}$	$\hat{\tau}(n)=\dfrac{S^2(n)}{\overline{X}(n)}$	D	备选的可变性的度量
偏度 $\upsilon=\dfrac{E[(X-\mu)^3]}{(\sigma^2)^{3/2}}$	$\hat{\upsilon}(n)=\dfrac{n^2}{(n-1)(n-2)}\dfrac{\sum_{i=1}^{n}[(X_i-\overline{X}(n))^3]/n}{(S^2(n))^{3/2}}$	C，D	对称性的度量

这些函数在某些情况下可以用于给出合适的分布类别的建议。对于一个对称的连续分布(如正态分布)，均值 μ 等于中值 $x_{0.5}$(对于对称的离散分布，总体均值和中值可能只是近似相等；参见第 4.2 节中关于中值的定义)。因此，如果估计 $\overline{X}(n)$和$\hat{x}_{0.5}(n)$“几乎相等”，这就说明该基本分布可能是对称的。人们应当记住，$\overline{X}(n)$和$\hat{x}_{0.5}(n)$是随机变量的观测值，它们的关系不必提供有关 μ 和 $x_{0.5}$间的真实关系准确信息。

方差系数 cv 有时也能提供关于连续分布形式的有用信息。特别是，指数分布的 cv=1，不管比例参数 β 是什么。因此，$\widehat{\mathrm{cv}}(n)$接近 1，则建议基本分布是指数的。对于伽马和韦布尔分布，当形状因子 α 小于、等于或大于 1 的时候，cv 分别大于、等于或者小于 1。更进一步，当 $\alpha>1$ 时，这些分布的形状与图 6.26 所示密度函数的形状类似，这意味着 cv<1。另一方面对数正态分布的密度函数形状总是与图 6.26 所示的相似，但是它的 cv 可以是任意的正实数。因此，如果基本分布(观测的直方图)具有这种形状，并且$\widehat{\mathrm{cv}}(n)>1$，对数正态分布可能是比伽马和韦布尔分布要好的模型。对于表 6.3 所示其余的分布，cv 并不是很有用[事实上，对于诸如 $U(-c, c)(c>0)$或者 $N(0, \sigma^2)$之类的分布，cv 并没有很好的定义，因为其均值为 0]。

对于离散分布，莱克塞斯(Lexis)比率 τ 起到连续分布方差系数同样的作用。我们发现莱克塞斯比率在鉴别泊松分布、二项分布，以及负二项分布时非常有用，因为这三个分布分别有$\tau=1$，$\tau<1$ 及 $\tau>1$(注意几何分布是负二项分布的一种特殊情形)。

偏度 υ 是关于分布对称性的一个测度量。对于对称的分布，如正态分布，$\upsilon=0$。如果 $\upsilon>0$(例如，对于指数分布 $\upsilon=2$)，分布向右侧偏；如果 $\upsilon<0$，分布向左侧偏。因此，估计的偏度估计 $\hat{\upsilon}(n)$(见 Joanes 与 Giu(1998))也可以用来判断真实基本分布的形状。我们的经验表明，很多在实际中遇到的分布都向右侧偏。

有可能定义分布参数的另外一个函数，称为峰度，它是分布“尾重”的度量[请参考文献 Kendall，Stuart 和 Ord(1987，第 107-108 页)]。不过，我们还没有发现峰度在区分分布时有特别大的作用。

6.4.2 直方图

对于连续的数据集，直方图本质上是对数据 X_1，X_2，…，X_n 分布对应的密度函数图的图形化估计(参见下面的讨论)。正如图 6.5 至图 6.16 所示的那样，密度函数在很多情况下都会有可辨认的形状。因此，密度函数的图形化估计对试图将数据建模为一个分布应该提供了一个很好的线索。

为了构造一个直方图，将数据覆盖的取值范围划分为 k 个不相交的相邻区间$[b_0, b_1)$，$[b_1, b_2)$，…，$[b_{k-1}, b_k)$。所有的区间都应当等宽，也就是 $\Delta b=b_j-b_{j-1}$，这样的话必须丢弃特别大或者特别小的 X_i 值，避免得到一个难以处理的直方图。对于 $j=1, 2, \cdots, k$，令 h_j 为落在第 j 个区间$[b_{j-1}, b_j)$中 X_i 的比例。最后，我们定义函数。

$$h(x)=\begin{cases}0, & x<b_0\\ h_j, & b_{j-1}\leqslant x<b_j, \quad 对于\ j=1,2,\cdots,k\\ 0, & b_k\leqslant x\end{cases}$$

将其绘成 x 的函数图(参见下面例 6.4 的直方图的例子)。h 的图是分段常数，然后将它和各种各样的分布函数只做形状比较(忽略位置和比例的差异)，看哪个分布的密度像直方图 h。

h 的形状像数据真实密度函数 f 的原因如下，令 X 为密度函数 f 的随机变量，那么 X 的分布与 X_i的分布一样。对于任何固定的 $j(j=1, 2, \cdots, k)$，某个数 $y\in(b_{j-1}, b_j)$，有

$$P(b_{j-1}\leqslant X<b_j)=\int_{b_{j-1}}^{b_j} f(x)\mathrm{d}x=\Delta b f(y)$$

第一个等式是由连续随机变量的定义得到，第二个等式从积分的中值定理可以得到。另一方面，X 落在第 j 个区间的概率近似为 h_j，也就是 $h(y)$的值。因此，

$$h(y)=h_j\approx\Delta b f(y)$$

这样，$h(y)$大致地正比于 $f(y)$；也就是，h 和 f 的形状大致相同(实际上，f 的估计就是将 h 除以常数 Δb 得到的)。

直方图可以应用于任意连续分布，并且提供了很直观的视觉观测。此外，可以很容易地看出，特定密度函数的直方图。然而还有些难点，就是缺乏确定区间数目 k(或者，等效地，它们的宽度 Δb)的权威性方法。

人们建议了一些经验方法来确定区间数 k{例如，Sturges 法[参见 Hoaglin，Mosteller 和 Tukey(1983，第 23-24 页)]，以及 Scott(1979)的正态逼近}。这些指南中最著名的可能是 Sturge 法，该方法称 k 应该按如下公式来确定：

$$k=\lfloor 1+\log_2 n\rfloor=\lfloor 1+3.322\log_{10} n\rfloor$$

然而，一般说来，我们不认为这种方法很有用(参见例 6.4)。我们建议尝试不同值的 Δb，挑选出使直方图“平滑”的那个最小值。这明显带有主观因素，也是使用直方图时最主要的问题。如果 Δb 选得过小，直方图就会有“锯齿”形，因为 h_j 的差异会很大。如果 Δb 选得太大，那么直方图就会有“块状”形，基本密度的形状就会被掩盖，因为数据过于聚集了。特别是，如果 Δb 太大，密度函数在 $x=0$ 或任何其他位置的尖峰会丢失。

正如已经说明的那样，直方图是对密度函数的估计(只是比例尺变化)。有很多其他的方法可由数据估计密度函数，其中一些相当复杂。我们建议有兴趣的读者参考 Wegman(1982，第 309-315 页)的综述文章以及 Silverman(1986)的书。

离散数据集的概率质量函数也可以通过直方图估计。对于通过数据估计的每个可能的 x_j，令 h_j 为 X_j 中等于 x_j 的比例。h_j 的纵轴高度作为 x_j 的函数，可以作一张图，将其同第 6.2.3 小节中离散分布的质量函数作形状对比。

对于离散数据集，h_j(它是随机变量)是 $p(x_j)$的无偏估计，其中 $p(x)$是该数据集真正(未知)的质量函数(参见习题 6.7)。然而，在这种情况下我们不需要对区间宽度和位置作任何主观的决策。

在某些情况下，直方图会有几个局部众数(或“峰”)，在这种情形，第 6.2 节讨论的标准分布不能提供足够精确的表示。图 6.31 给出了这样一个例子，它代表了一年内搜集到的维修某机器所需要的时间(连续的)。该机器有两类故障：大多数是小故障，对应的维修时间很小；这种情况对应图 6.31 所示的左峰；小比例的故障是大故障，需要大的维修时间，因为备用零件需要订购。这导致了图 6.31 所示的右峰。如果所观测的维修时间可以分为两种情形(对应于小修和大修)，p_j 代表了情形 $j(j=1, 2)$观测所占的比例，那么应用在第 6.4 节中讨论过的方法，密度 $f_j(x)$是对情形 j 的观测的拟合。因此，整个维修时间密度 $f(x)$为：

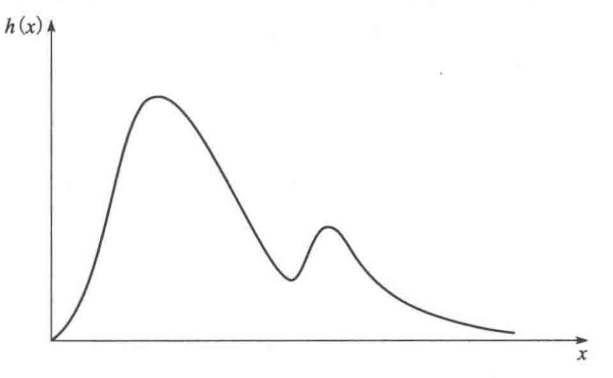


图 6.31 对应于密度函数的直方图，其中有两个局部众数

$$f(x) = p_1 f_1(x) + p_2 f_2(x)$$

在仿真中，使用组合法(参见第 8.2.2 小节)，随机维修时间可以由 $f(x)$产生。对于有多个众数的直方图，建模的其他方法参见第 6.9 节和习题 6.8。

6.4.3 分位数求和与盒形图

分位数求和[例如，参考 Tukey(1970)]是对样本的综合分析方法，用于判断潜在概率密度函数或概率质量函数是对称的或者向左、右偏移。它可以用于连续或者离散数据集；然而为表述方便，只介绍连续的例子。

假设 $F(x)$为连续随机变量分布函数。更进一步假设当 $0<F(x)<1$ 时，$F(x)$为连续严格增函数[这意味着如果 $x_1<x_2$ 并且 $0<F(x_1)\leqslant F(x_2)<1$，那么 $F(x_1)<F(x_2)$]。对于 $0<q<1$，$F(x)$的 q 分位数是指 x_q，使得 $F(x_q)=q$。如果令 F^{-1}指 $F(x)$的反函数，那么 $x_q=F^{-1}(q)$\{F^{-1}具有性质 $F[F^{-1}(x)]=F^{-1}[F(x)]=x$\}。这里我们对中值 $x_{0.5}$、上下四分位数 $x_{0.25}x_{0.75}$、上下八分位数 $x_{0.125}x_{0.875}$ 特别感兴趣。样本 X_1，X_2，…，X_n 的分位数求和结构如表 6.6 所示。

表 6.6 对于样本 X_1，X_2，…，X_n 的分位数求和结构

分位数	深度	样本值		中点
中值	$i=(n+1)/2$	$X_{(i)}$		$X_{(i)}$
四分位数	$j=(\lfloor i \rfloor+1)/2$	$X_{(j)}$，	$X_{(n-j+1)}$	$[X_{(j)}+X_{(n-j+1)}]/2$
八分位数	$k=(\lfloor j \rfloor+1)/2$	$X_{(k)}$，	$X_{(n-k+1)}$	$[X_{(k)}+X_{(n-k+1)}]/2$
极值	1	$X_{(1)}$，	$X_{(n)}$	$[X_{(1)}+X_{(n)}]/2$

在表 6.6 中，如果深度下标 l(l 等于 i，j 或 k)在整数 m 和 $m'=m+1$ 之间，那么 $X_{(l)}$定义为 $X_{(m)}$和 $X_{(m')}$的平均值。$X_{(i)}$是对中值的估计，$X_{(j)}$和 $X_{(n-j+1)}$是对四分位数的估计，$X_{(k)}$和 $X_{(n-k+1)}$是对八分位数的估计。如果 X_i 潜在分布是对称的，那么四个中点应当近似相同。另一方面，如果 X_i 潜在分布向右(左)偏移，那么四个中点(表中从上到下)应当依次增加(减小)。

盒形图是分位数的图形化表示(参考图 6.33)。50%的观测值落在箱体[$x_{0.25}$，$x_{0.75}$]的垂直边界内。

下面两个例子说明了如何使用第 6.4 节中的方法。

例 6.4 为了对免下车方便驾车者的银行设备开发一个仿真模型，搜集了车辆到达的模型。在一个固定的 90min 区间上，220 量车到达。对于 $i=1$，2，…，219，记第 i 辆和

$i+1$ 辆车到达的间隔时间为 X_i。表 6.7 列出了 $n=219$ 个间隔时间，并按升序排列。对连续 6 个 15min 内到达的车辆记数，发现它们近似相同，表明在整个 90min 内，车辆到达的速率近似为常数。此外，汽车每次到达一辆，没有理由相信非连续区间内车辆到达的数量相互独立。因此，从理论上看，我们认为间隔时间是服从指数分布的。为了证明这个假设，我们首先观察表 6.8 所示的统计数据。因为 $\overline{X}(219)=0.399>0.270=\hat{x}_{0.5}(219)$，$\hat{v}(219)=1.458$，这表明潜在分布向右偏斜，而不是对称的。此外，$\widehat{\text{cv}}(219)=0.953$，接近指数分布的理论值 1。下一步我们做了三个不同数据直方图，如图 6.32 所示，在每个例子中 $b_0=0$，Δb 分别为 0.050，0.075，0.100。$\Delta b=0.100$ 时指标图平滑曲线最像指数分布密度(注意按照 Sturges 规则，$k=8$，$\Delta b=0.250$，导致数据过度聚集)。最后，图 6.33 给出了到达时间的分位数统计和盒形图。递增的中点和盒形图右侧延长的部分再次肯定了潜在分布为指数分布。总之，从理论和经验上看，假设到达时间是指数分布。

表 6.7 按照升序排列的 $n=219$ 个间隔时间 (单位：min)

0.01	0.06	0.12	0.23	0.38	0.53	0.88
0.01	0.07	0.12	0.23	0.38	0.53	0.88
0.01	0.07	0.12	0.24	0.38	0.54	0.90
0.01	0.07	0.13	0.25	0.39	0.54	0.93
0.01	0.07	0.13	0.25	0.40	0.55	0.93
0.01	0.07	0.14	0.25	0.40	0.55	0.95
0.01	0.07	0.14	0.25	0.41	0.56	0.97
0.01	0.07	0.14	0.25	0.41	0.57	1.03
0.02	0.07	0.14	0.26	0.43	0.57	1.05
0.02	0.07	0.15	0.26	0.43	0.60	1.05
0.03	0.07	0.15	0.26	0.43	0.61	1.06
0.03	0.08	0.15	0.26	0.44	0.61	1.09
0.03	0.08	0.15	0.26	0.45	0.63	1.10
0.04	0.08	0.15	0.27	0.45	0.63	1.11
0.04	0.08	0.15	0.28	0.46	0.64	1.12
0.04	0.09	0.17	0.28	0.47	0.65	1.17
0.04	0.09	0.18	0.29	0.47	0.65	1.18
0.04	0.10	0.19	0.29	0.47	0.65	1.24
0.04	0.10	0.19	0.30	0.48	0.69	1.24
0.05	0.10	0.19	0.31	0.49	0.69	1.28
0.05	0.10	0.20	0.31	0.49	0.70	1.33
0.05	0.10	0.21	0.32	0.49	0.72	1.38
0.05	0.10	0.21	0.35	0.49	0.72	1.44
0.05	0.10	0.21	0.35	0.50	0.72	1.51
0.05	0.10	0.21	0.35	0.50	0.74	1.72
0.05	0.10	0.21	0.36	0.50	0.75	1.83
0.05	0.11	0.22	0.36	0.51	0.76	1.96
0.05	0.11	0.22	0.36	0.51	0.77	
0.05	0.11	0.22	0.37	0.51	0.79	
0.06	0.11	0.23	0.37	0.52	0.84	
0.06	0.11	0.23	0.38	0.52	0.86	
0.06	0.12	0.23	0.38	0.53	0.87	

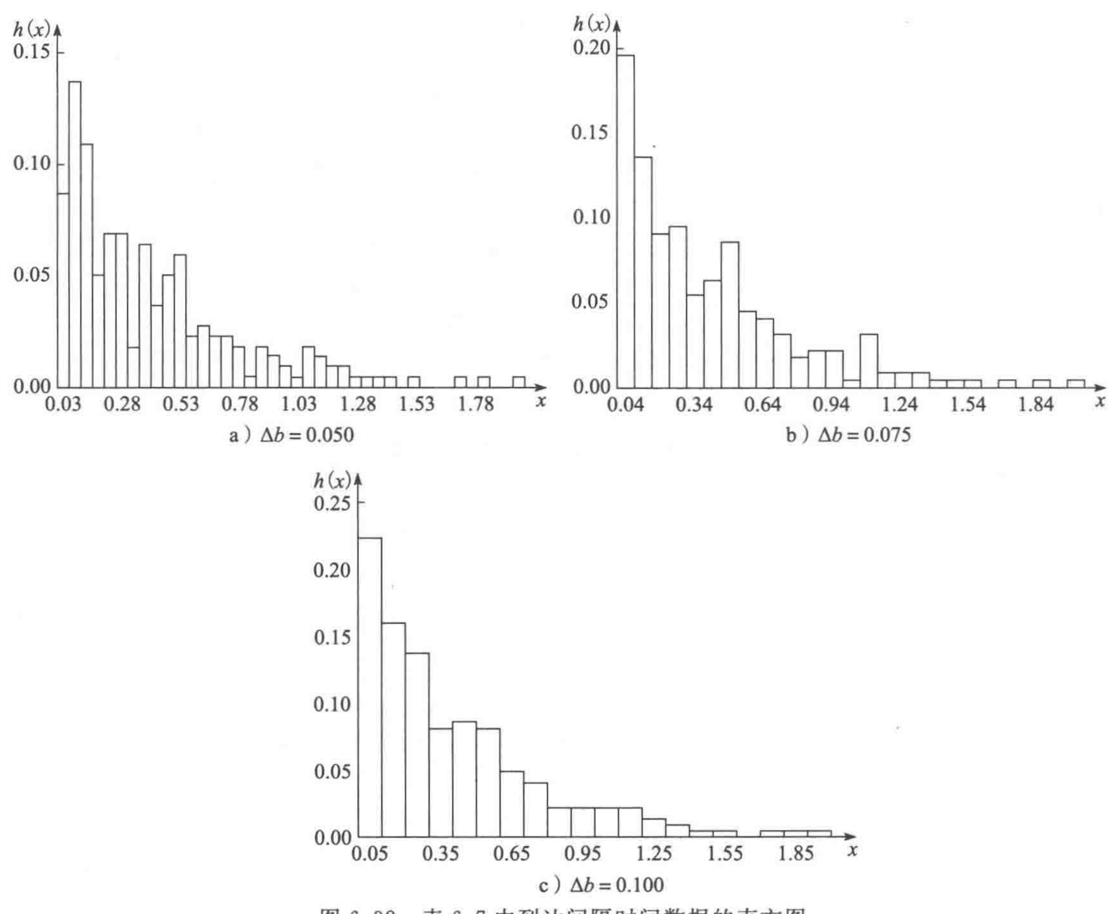


图 6.32　表 6.7 中到达间隔时间数据的直方图

表 6.8　到达间隔时间数据的求和统计

求和统计	值
最小	0.010
最大	1.960
均值	0.399
中值	0.270
方差	0.144
方差系数	0.953
偏斜度	1.458

分位数	深度	样本均值		中点
中值	110	0.270		0.270
分位数	55.5	0.100	0.545	0.323
45°直线	28	0.050	0.870	0.460
极值	1	0.010	1.960	0.985

图 6.33　到达时间间隔的分位数统计求和与盒形图

例 6.5 表 6.9 给出了三年时间段内，每周库存所需零件数据的 $n=156$ 个值以及计数，并按照升序排列。我们没有给出所有个体值，而是频率统计；59 个 X_i 等于 0，26 个 X_i 等于 1，等等。这些数据的概要统计和直方图如表 6.10 和图 6.34 所示。由于雷氏比例 $\hat{\tau}(156)=2.795$，模型不像是二项分布或者泊松分布。此外，偏斜值 $\hat{\nu}(156)=1.655$ 为正且很大，也排除了对称的离散正态分布的可能性。因此，可能的离散模型为几何分布或者是负二项分布。当 $s=1$ 时前者是后者的一种特殊形式。然而，图 6.34 所示直方图单调递减(图 6.21 所示的质量函数)，我们假定需求数据服从几何分布。

表 6.9　按照升序排列的 $n=156$ 个需求量值和计数

0(59),	1(26),	2(24),	3(18),	4(12),
5(5),	6(4),	7(3),	9(3),	11(2)

表 6.10　需求量数据的求和统计

求和统计	值
最小	0.000
最大	11.000
均值	1.891
中值	1.000
方差	5.285
方差系数	2.795
偏斜度	1.655

◀

6.5　活动Ⅱ：参数估计

当我们在活动Ⅰ中假定了一个或多个备选分布类别后，我们必须指定参数值，以完全确定分布，用于仿真。独立同分布数据 X_1，X_2，…，X_n 可用于帮助我们假设分布，并且用于参数估计。当以这种方式直接使用数据估计参数时，我们认为这是从数据中估计参数。

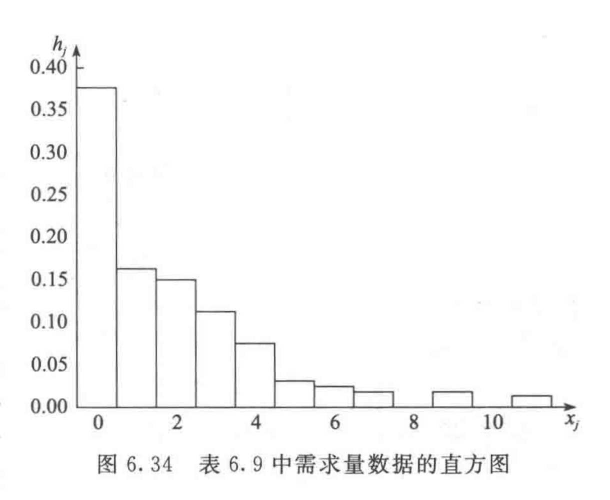


图 6.34　表 6.9 中需求量数据的直方图

估计是数据的一个样本函数。对于给定分布的特定参数，有很多方法指定估计的形式，也有一些评估估计质量的别的方法。我们只考虑其中一个类型，最大似然估计(maximum-likelihood estimator，MLE)。这是基于三点考虑：①MLE 具有很多其他估计例如最小二乘估计、无偏估计、矩估计等不具有的良好性质；②使用 MLE 对于判断 x^2 优良度检验(参见 6.6.2 小节)很重要；③从直观上讲 MLE 的中心观点具有很强的说服力。

离散情况下的 MLE 最容易理解。假设我们对数据假定了一个分布类型，具有参数 θ。令 $p_\theta(x)$表示这个分布的概率质量函数，那么参数 θ 是定义的一部分。假设我们已经有了独立同分布观测数据 X_1，X_2，…，X_n，可以按照下面的方式定义似然函数 $L(\theta)$：

$$L(\theta) = p_\theta(X_1)p_\theta(X_2)\cdots p_\theta(X_n) \tag{6.1}$$

因为数据是相互独立的(参见第 4.2 节)，$L(\theta)$正是联合概率质量函数。$L(\theta)$给出了 θ 为未知参数情况下获得观测数据的概率(似然)。将未知参数 θ 的 MLE 记作$\hat{\theta}$，定义为最大化

$L(\theta)$的 θ 值，也就是对于所有可能的 θ，有 $L(\hat{\theta})\geqslant L(\theta)$。因此，$\hat{\theta}$"最好的解释"了收集的数据。在连续情况下，MLE 没有这么直观的解释，因为连续随机变量等于任意一个固定数的概率都是 0[参见习题 6.26 和 Breiman(1973，第 67-68 页)，其中有关于 MLE 在连续情况下的直观解释]。然而至少可以将 MLE 定义为和离散情况下类似。如果令 $f_\theta(x)$ 为假设密度函数(再次假设这里只有一个未知参数 θ)，似然函数定义为：

$$L(\theta) = f_\theta(X_1)f_\theta(X_2)\cdots f_\theta(X_n) \tag{6.2}$$

θ 的 MLE $\hat{\theta}$定义为 θ 取所有可能值中最大化 $L(\theta)$的 θ 值。下面的两个例子说明前面的例 6.4 和例 6.5 如何计算假设分布的 MLE。

例 6.6 对于指数分布，$\theta=\beta(\beta>0)$，以及 $f_\beta(x)=(1/\beta)e^{-x/\beta}$，$x\geqslant 0$，似然函数为：

$$L(\beta)=\left(\frac{1}{\beta}e^{-X_1/\beta}\right)\left(\frac{1}{\beta}e^{-X_2/\beta}\right)\cdots\left(\frac{1}{\beta}e^{-X_n/\beta}\right)$$

$$=\beta^{-n}\exp\left(\frac{1}{\beta}\sum_{i=1}^{n}X_i\right)$$

我们的目的是，对于所有的 $\beta>0$，寻找最大化 $L(\beta)$的 β 值。这个任务如果使用 $L(\beta)$的对数来代替直接对 $L(\beta)$来做，将更容易解决。因此，定义对数似然函数为：

$$l(\beta) = \ln L(\beta) = -n\ln\beta - \frac{1}{\beta}\sum_{i=1}^{n}X_i$$

由于对数函数严格递增，最大化 $L(\beta)$等同于最大化 $l(\beta)$，后者容易得多；也就是说，$\hat{\beta}$最大化 $L(\beta)$当且仅当$\hat{\beta}$最大化 $l(\beta)$。可以使用标准微分计算方法来最大化 $l(\beta)$，办法是令其导数为 0，求解 β。也就是，

$$\frac{dl}{d\beta} = \frac{-n}{\beta} + \frac{1}{\beta^2}\sum_{i=1}^{n}X_i$$

上式为 0 当且仅当 $\beta=\sum_{i=1}^{n}X_i/n=\overline{X}(n)$。为保证 $\beta=\overline{X}(n)$是 $l(\beta)$的极大值(而不是极小值或者拐点)，一个充分(但不是必要)条件是$\frac{d^2l}{d\beta^2}$在 $\beta=\overline{X}(n)$时为负。而

$$\frac{d^2l}{d\beta^2} = \frac{n}{\beta^2} - \frac{2}{\beta^3}\sum_{i=1}^{n}X_i$$

很容易看出，X_i 为正，则上式在 $\beta=\overline{X}(n)$时为负。因此 β 的 MLE 为$\hat{\beta}=\overline{X}(n)$。注意到 MLE 的结果在这里是十分自然的，因为 β 为假设分布的均值，而 MLE 为样本均值。对于例 6.4 中的数据，$\hat{\beta}=\overline{X}(219)=0.399$。◀

例 6.7 例 6.5 中的离散数据假设来自一个几何分布。这里，对于 $x=0$，1，…，$\theta=p(0<p<1)$，$p_p(x)=p(1-p)^x$。似然函数为：

$$L(p) = p^n(1-p)^{\sum_{i=1}^{n}X_i}$$

它也能服从对数变换，得到：

$$l(p) = \ln L(p) = n\ln p + \sum_{i=1}^{n}X_i\ln(1-p)$$

对 $l(p)$求导，得到：

$$\frac{dl}{dp} = \frac{n}{p} - \frac{\sum_{i=1}^{n}X_i}{1-p}$$

当且仅当 $p=1/[\overline{X}(n)+1]$时，上式等于 0。为确保这是个极大值，注意到对于任意有效的 p，有：

$$\frac{\mathrm{d}^2 l}{\mathrm{d}p^2} = -\frac{n}{p^2} - \frac{\sum_{i=1}^{n} X_i}{(1-p)^2} < 0$$

因此 p 的 MLE 为 $\hat{p}=1/[\overline{X}(n)+1]$，这个结果也很直观(参见习题 6.9)。对于例 6.5 中的需求量数据，$\hat{p}=0.346$。 ◀

上面两个例子说明了求解 MLE 的两个重要的实际工具，即对数似然函数的应用，以及令其导数(关于被估计参数)为 0 以得到 MLE。虽然这些工具在求解 MLE 时经常用到，读者应该注意到，求解 MLE 并不总是令导数为 0 并容易地求解$\hat{\theta}$这样简单的事情。对于某些分布，对数似然函数和微分都不可用；或许最熟悉的例子就是均匀分布(参见习题 6.10)。对于其他的分布，这两个工具都可用，但是求解 $\mathrm{d}l/\mathrm{d}\theta=0$ 不能够通过简单的代数计算得到，而必须使用数值方法；伽马分布、韦布尔分布，以及贝塔分布都是这种一般情况的(多参数)例子。对于用于求解各种分布 MLE 的方法的例子，我们建议读者参考文献 Breiman(1973，第 65-84 页)。

MLE 具有一些很理想的统计性质，其中一些如下所示[参见 Breiman(1973，第 85-88 页)以及 Kendall 和 Stuart(1973，第 18 章)]：

(1) 对于大多数常用的分布，MLE 是唯一的；也就是说，对于其他任何的 θ 值，$L(\hat{\theta})$ 都严格大于 $L(\theta)$；

(2) 虽然 MLE 不必是无偏的，一般情况下，$\hat{\theta}$的渐近分布($n\to+\infty$)的均值等于 θ(参见下面的性质 4)；

(3) MLE 具有不变性(invariant)，也就是，如果对于某个函数 h，$\varphi=h(\theta)$，那么 φ 的 MLE 为 $h(\hat{\theta})$(无偏性不是不变性)。例如，expo(β)随机变量的方差为 β^2，因此方差的 MLE 为$[\overline{X}(n)]^2$；

(4) MLE 是渐近正态分布的；也就是说，$\sqrt{n}(\hat{\theta}-\theta)\xrightarrow{\mathscr{D}}N(0,\ \delta(\theta))$，其中，$\delta(\theta)=-n/E(\mathrm{d}^2 l/\mathrm{d}\theta^2)$(假设 X_i 具有假定的分布，期望是关于 X_i 的函数)$\xrightarrow{\mathscr{D}}$表示分布的收敛。此外，如果$\tilde{\theta}$是满足$\sqrt{n}(\hat{\theta}-\theta)\xrightarrow{\mathscr{D}}N(0,\ \sigma^2)$的任意其他估计，那么 $\delta(\theta)\leqslant\sigma^2$(MLE 称为最佳渐近正态估计)；

(5) MLE 是强一致的，也就是说，$\lim\limits_{n\to+\infty}\hat{\theta}=\theta$，w. p. 1。

上述性质以及其他性质的证明需要附加温和的“正则性”假设，参见 Kendall 和 Stuart(1979)。

性质(4)特别有趣，因为它使我们能够为 θ 建立一个近似的置信区间。如果按照上述性质(4)定义 $\delta(\theta)$，那么 $n\to+\infty$时，可以证明

$$\frac{\hat{\theta}-\theta}{\sqrt{\delta(\hat{\theta})/n}}\xrightarrow{\mathscr{D}}N(0,1)$$

因此，对于很大的 n，θ 的近似 $100(1-\alpha)\%$置信区间为：

$$\hat{\theta}\pm z_{1-\alpha/2}\sqrt{\frac{\delta(\hat{\theta})}{n}} \tag{6.3}$$

例 6.8 为几何分布的参数 p 构造一个 90%的置信区间，采用例 6.5 中的数据来确定它。很容易看出，

$$E\left[\frac{\mathrm{d}^2 l}{\mathrm{d}p^2}\right] = -\frac{n}{p^2} - \frac{n(1-p)/p}{(1-p)^2} = -\frac{n}{p^2(1-p)}$$

因此，$\delta(p)=p^2(1-p)$，且对于大的 n，p 的近似 90%的置信区间为：

$$\hat{p}\pm 1.645\sqrt{\frac{\hat{p}^2(1-\hat{p})}{n}}$$

对于例 6.5 中的数据，得到 0.346 ± 0.037。 ◀

这也提供了一种检查仿真输出性能度量对特定输入参数的灵敏度如何的方法。可将仿真运行在譬如说式(6.3)的置信区间 θ 的左端点、中心点($\hat{\theta}$)、右端点。如果性能度量对该范围内 θ 的值呈现出不敏感，那么我们就会有信心认为，对于我们的目的来说，我们有 θ 的相当好的估计。另一方面，如果仿真呈现出对 θ 是敏感的，我们也许要找到一个 θ 的更好的估计，这往往需要搜集更多的数据。

上述问题的一般形式可以描述如下。仿真模型的性能度量依赖于输入概率分布和其相关参数的选择。当我们选择这些分布用于一个仿真模型时，我们通常并不绝对肯定地知道使用这些分布是否正确，这种缺乏完整知识导致了我们所说的模型不确定性。此外，假设选择了特定输入分布，我们一般也不完全肯定对这些分布要用什么参数，我们称这为参数不确定性(参数通常从观测数据来估计或者根据专家经验主观给定)。术语“输入模型不确定性”通常指这两个问题的全部。理想情况下，我们很希望有一个构建仿真性能评价的置信区间的方法，该方法同时考虑仿真模型的抽样变化(见第 9 章)和输入模型的不确定性。Henderson(2003)认为该方法基于统计过程很容易被仿真实践者所理解，相对来说，也很容易实施与提高计算效率。

Barton(2012)和 Henderson(2003)中对输入模型不确定性给出了很好的介绍，其中讨论了解决这个问题的一系列方法，包括下面这些：

- 贝叶斯模型平均值方法[Chick(2001)，以及 Zouaoui 和 Wilson(2003，2004)]
- Delta 方法[Cheng 和 Holland(1997，1998，2002)]
- 元模型辅助的自举方法[关于自举重采样的讨论，参见 Barton 等(2013)第 12 章，以及 Cheng(2006)和 Efron 和 Tibshirani(1993)]
- 非参数的自举方法[参见 Barton 和 Schruben(1993，2001)]
- 基于随机效用模型的快速方法[Ankenman 和 Nelson (2012)]

但是，这些方法中的绝大多数在理论上是复杂的，而且在实际上通过假设很难满足要求[见 Barton(2012)和 Barton 等(2013)]。例如，贝叶斯模型平均值方法和 Delta 方法都预先假设分布(不是参数值)族(族系列)能够很好地描述系统随机性来源，但是，在绝大多数实际的应用里这不可能是真的。

到目前为止，我们只对具有单一未知参数的分布明确地进行了处理。如果一个分布具有多个参数，可以很自然地定义这些参数的 MLE。例如，伽马分布具有两个参数(α 和 β)，从而似然函数定义为：

$$L(\alpha,\beta)=\frac{\beta^{-n\alpha}\ \left(\prod_{i=1}^{n}X_i\right)^{\alpha-1}\exp\left[-(1/\beta)\sum_{i=1}^{n}X_i\right]}{\left[\Gamma(\alpha)\right]^n}$$

α 和 β 的未知值的 MLE 的$\hat{\alpha}$和$\hat{\beta}$定义为联合最大化 $L(\alpha,\beta)$的 α 和 β 值[求解$\hat{\alpha}$和$\hat{\beta}$方法如下：令 $l(\alpha,\beta)=\ln L(\alpha,\beta)$，试解方程 $\partial l/\partial\alpha=0$ 与 $\partial l/\partial\beta=0$]。前面列出的 MLE 的类似性质在多参数情形下也成立。不幸的是，当有多个参数时求解 MLE 的过程通常是相当困难的(正态分布很例外)。

对于第 6.2.2 小节和第 6.2.3 小节中的每个分布(约翰逊 S_B、约翰逊 S_u，以及三角分布例外)，我们给出了 MLE 求解公式或者求解他们的数值方法。对于伽马 MLE，表 6.21 可以用于标准线性插值。对于贝塔 MLE，可以使用表 6.22；方法可以是，或者从相应于 G_1 与 G_2 的最靠近的表格值中直接挑选($\hat{\alpha}_1$，$\hat{\alpha}_2$)，或者设计一个二维插值方法。

6.6　活动Ⅲ：判断拟合分布的代表性

在活动Ⅰ和Ⅱ中选择了一个或多个概率分布以拟合观测数据，接下来要认真检查该分布是否代表了数据潜在的真实分布。如果这些分布中有一些具有“代表性”，还必须判断哪个提供了最好的拟合。一般情况下，拟合的分布没有是完全正确的。真正试图要做的事

情是为模型的努力目标确定一个足够精确的分布。

在本节，讨论启发式方法与拟合优良度假设检验法两种方法，以确定所拟合的分布的“质量”。

6.6.1　启发式方法

下面将讨论五个启发式或图形化方法，以用于将拟合分布同真正的基本分布作比较；一些其他的方法可以在 Averill M. Law& Associates(2013)中找到。

密度直方图和频率比较

对于连续数据，将 $\Delta b\,\hat{f}(x)$画在直方图 $h(x)$上并寻找相似性就可以做出密度直方图[回忆一下，$h(x)$下的面积是 Δb]。频率比较是另一种将数据的直方图同拟合分布的密度函数$\hat{f}(x)$进行比较的图形化方法。令$[b_0, b_1)$，$[b_1, b_2)$，…，$[b_{k-1}, b_k)$为一组 k 个直方图区间，每个宽度为 $\Delta b=b_j-b_{j-1}$。令 h_j 为观测到的 X_i 落在第 j 个区间$[b_{j-1}, b_j)$上的比例，如果拟合分布事实上为真实分布，令 r_j 为 n 个观测值落在第 j 个区间上的期望比例；即 r_j 由下式给出(参见习题 6.13)：

$$r_j=\int_{b_{j-1}}^{b_j}\hat{f}(x)\mathrm{d}x \tag{6.4}$$

那么，对于 $j=1, 2, \cdots, k$，将第 j 个直方图区间上的 h_j 和 r_j 一起绘图，就进行了频率比较。

对于离散数据，频率比较就是将数据的直方图与拟合分布的质量函数$\hat{p}(x)$进行图形比较。令 h_j 为所观测到的 X_i 中等于x_j 的个数比例，如果拟合分布事实上为真实分布，令 r_j 为 n 个观测值中等于 x_j 的个数的期望比例，即 $r_j=\hat{p}(x_j)$。那么，对于所有相关的 x_j 值，相对 x_j 绘制 h_j 和 r_j 的图，就进行了频率比较。

无论是连续还是离散的情况，如果拟合分布很好地代表了数据的真实的基本分布(如果样本大小 n 足够大)，则 r_j 与 h_j 应该很接近一致。

例 6.9　对于例 6.4 中的到达间隔时间数据，假设一个指数分布，并且例 6.6 中 MLE $\hat{\beta}=0.399$。因此拟合分布的密度为：

$$\hat{f}(x)=\begin{cases}2.506\mathrm{e}^{-x/0.399}, & x=0,1,2,\cdots\\ 0, & \text{其他}\end{cases}$$

对于图 6.32c 所示的直方图，在图 6.35 中给出了密度直方图。◀

例 6.10　例 6.5 中的需求量数据假设来自几何分布，参数 p 的 MLE 在例 6.7 中求得为$\hat{p}=0.346$。因此，拟合分布的质量函数为：

$$\hat{p}(x)=\begin{cases}0.346\times(0.654)^x, & x\geqslant 0\\ 0, & \text{其他}\end{cases}$$

对于图 6.35 所示的直方图，对于 $j=1, 2, \cdots, 12$，$r_j=\hat{p}(x_j)=\hat{p}(j-1)$，图 6.36 中给出了频率比较，其中 h_j 为白柱，r_j 为灰柱。再一次，除了 $x_2=1$ 外，一致性是好的。◀

分布函数差图

密度直方覆盖图可以看做将拟合分布的个体概率与真实的基本分布的个体概率进行对比。我们也可以进行累计概率(分布函数)的图形化比较。定义一个(新的)经验分布函数 $F_n(x)$为：

$$F_n(x)=\frac{X_i\leqslant x\text{ 的个数}}{n} \tag{6.5}$$

它就是观测数据中小于或等于 x 的比例。然后，将$\hat{F}(x)$(拟合分布的分布函数)同 $F_n(x)$绘制在一张图中，并寻找相似性。然而，分布函数的特征一般没有密度函数或质量函数明显。事实上，很多分布函数都具有某类“S”的形状，目视“S”状曲线之间的差异或相似性比较困难。因此将分布函数差图定义为$\hat{F}(x)$和 $F_n(x)$之间的差值图。如果拟合分布是

足够好的拟合，样本量是无限的，那么此图将是一条高度为 0 的水平线。因此，距离这条线的垂直偏差越大，拟合的质量越差。很多在绝对意义下很差的拟合分布在数据区间的低端都有很大的偏差。

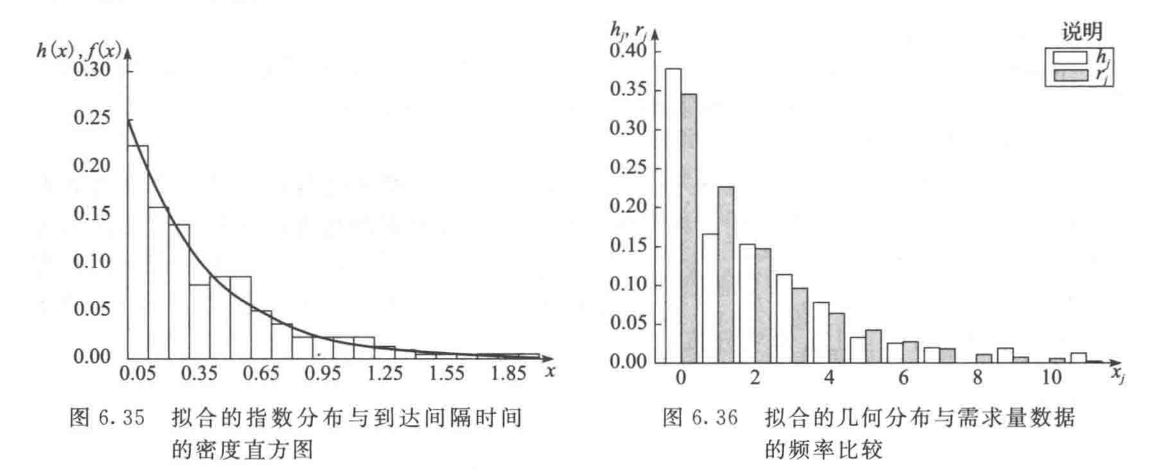


图 6.35 拟合的指数分布与到达间隔时间的密度直方图

图 6.36 拟合的几何分布与需求量数据的频率比较

例 6.11 图 6.37 中给出了例 6.4 中到达间隔时间数据与拟合的指数分布的分布函数差图。该图表明，除观测数据的低端区域外，其他部分都拟合得很好(水平虚线为误差限界，它依赖于样本大小 n。如果差值图超过了这些线，那么这强烈意味着拟合效果很差。这些误差限界从 6.7 节 35 000 组数据中得到的)。◀

例 6.12 图 6.38 中给出了例 6.5 中需求量数据和拟合的几何分布的分布函数差图，该图表明除了观测数据低端区域外，其他部分都拟合得很好。

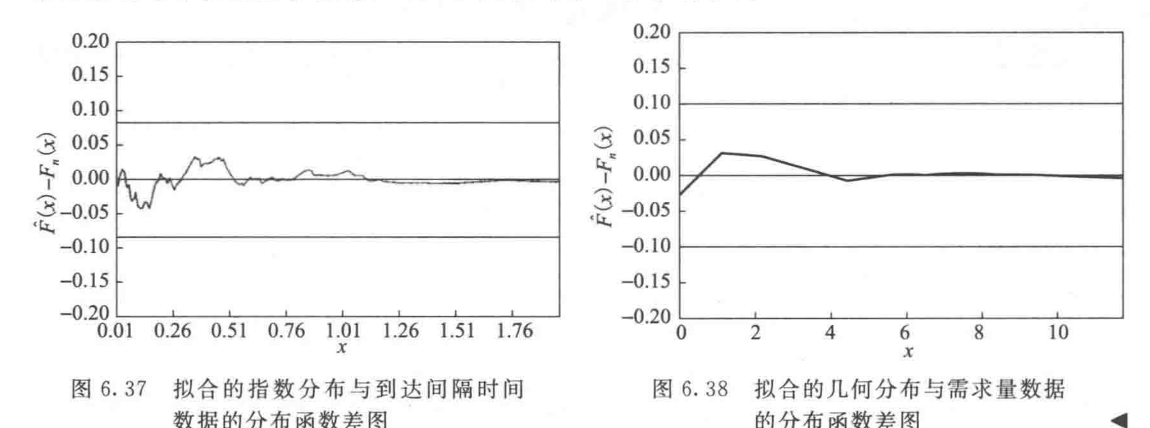


图 6.37 拟合的指数分布与到达间隔时间数据的分布函数差图

图 6.38 拟合的几何分布与需求量数据的分布函数差图 ◀

概率图

概率图可以看做另一种图形化比较方法，它将观测数据 X_1，X_2，…，X_n 的真实分布的估计与拟合分布的分布函数作对比。有很多种(以及用法)概率图，这里我们只讨论其中的两种；其他的讨论参见 Barnett(1975)、Hahn 和 Shapiro(1994)、Wilk 和 Gnanadesikan (1968)。

正如第 6.2.4 小节中一样，令 $X_{(i)}$ 为 X_j 中第 i 个最小的，有时称作 X_j 的第 i 阶统计。随机变量 X 的分布函数 $F(X)$ 的一个合理的估计是 $F_n(X)$，它是由方程式(6.5)定义的。注意到 $F_n(X_{(i)})=i/n$。然而，为绘制概率图，$F_n(X_{(n)})=1$ 这有些不方便，也就是说，对于有限的 x 值，经验分布函数等于 1(参见习题 6.14)。因此，这里将使用下面的经验分布函数：

$$\widetilde{F}_n(X_{(i)}) = F_n(X_{(i)}) - \frac{0.5}{n} = \frac{i-0.5}{n}$$

显然，对于中等大小的 n，这个调整是相当小的。也有人建议使用其他形式的调整，例如 $i/(n+1)$。然后一个直接的方法就是绘制 n 个点 $(X_{(1)}, 0.5/n)$，$(X_{(2)}, 1.5/n)$，…，$(X_{(n)}, (n-0.5)/n)$ 的图，并将这个结果同考虑作为数据的模型的分布函数图对比，寻找相似之处。然而，正如前文所述，很多分布函数都具有某类“S”形状，目视“S”状曲线之间的相似性或差异是困难的。然而，我们大多数都可以分辨出一组绘制点靠近直线是多还是少，概率图方法将比较分布函数的问题简化为寻找直线的问题。

对于 $i=1, 2, \cdots, n$，令 $q_i=(i-0.5)$，因此 $0<q_i<1$。对于任何连续数据集（参见习题 6.15），分位数-分位数(Q-Q)图（分位数的定义参见第 6.4.3 小节）是这样一个图，对于 $i=1, 2, \cdots, n$，将拟合（模型）的分布函数 $\hat{F}(X)$ 的 q_i 分位数，即 $x_{q_i}^M=\hat{F}^{-1}(q_i)$，与样本分布函数 $\widetilde{F}_n(X)$ 的 q_i 分位数，即 $x_{q_i}^S=\widetilde{F}_n^{-1}(q_i)=X_{(i)}$ 绘制在一张图中。图 6.39 给出了 Q-Q 图的定义，为方便起见，其中，将 $\widetilde{F}_n(x)$ 表示为平滑曲线。与每个坐标 q 值对应的是两个分位数 x_q^M 和 x_q^S。

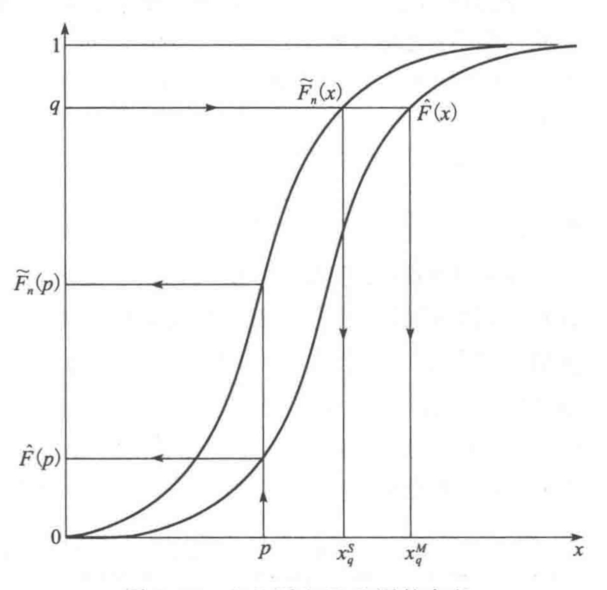


图 6.39　Q-Q 图和 P-P 图的定义

如果 $\hat{F}(x)$ 同是与真实的基本分布 $F(x)$ 相同的分布，而且样本数量很大，那么 $\hat{F}(x)$ 和 $\widetilde{F}_n(x)$ 将紧挨在一起，且 Q-Q 图必将是近似的一直线，截距为 0，斜率为 1。即使 $\hat{F}(x)$ 是正确的分布，对小到中等的样本数量来说，也会出现偏离直线的情况。

概率-概率图（P-P 图）是模型概率 $\hat{F}(x_{(i)})$ 对样本概率 $\overline{F}_n(x_{(i)})=q_i$ $(i=1, 2, \cdots, n)$ 的图；它对连续和离散数据集两者都适用。该定义也在图 6.39 中表示了。对应于横坐标的每个 p 值有两个概率 $\hat{F}(p)$ 和 $\widetilde{F}_n(p)$。如果 $\hat{F}(p)$ 和 $\widetilde{F}_n(p)$ 十分接近，那么 P-P 图也将近似为一直线，截距为 0，斜率为 1。

Q-Q 图会放大模型分布函数 $\hat{F}(x)$ 尾部与样本分布函数 $\widetilde{F}_n(x)$ 尾部之间存在的差异，而 P-P 图会放大 $\hat{F}(x)$ 中部与 $\widetilde{F}_n(x)$ 中部之间的差异。图 6.40 所示的分布函数右尾之间的差异被 Q-Q 图放大了，而 P-P 图不会。另一方面，图 6.41 所示的两个分布函数的“中部”的差异被 P-P 图放大了。

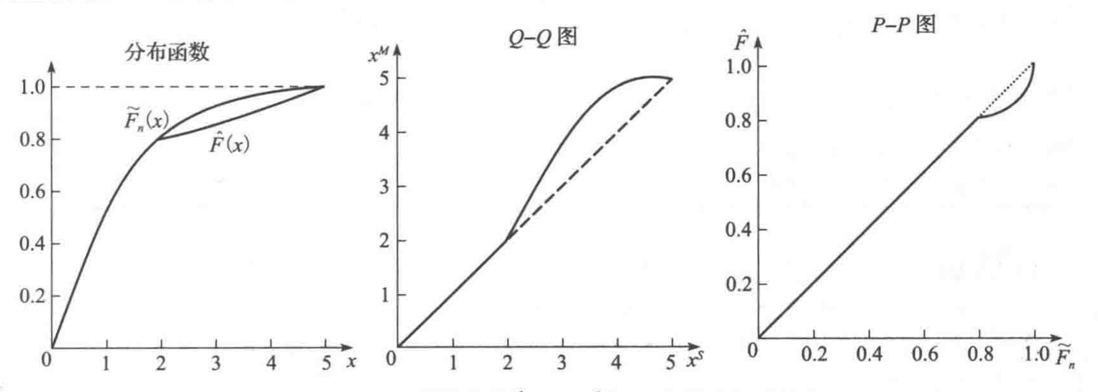


图 6.40　Q-Q 图放大的 $\hat{F}(x)$ 与 $\widetilde{F}_n(x)$ 右尾之间的差异

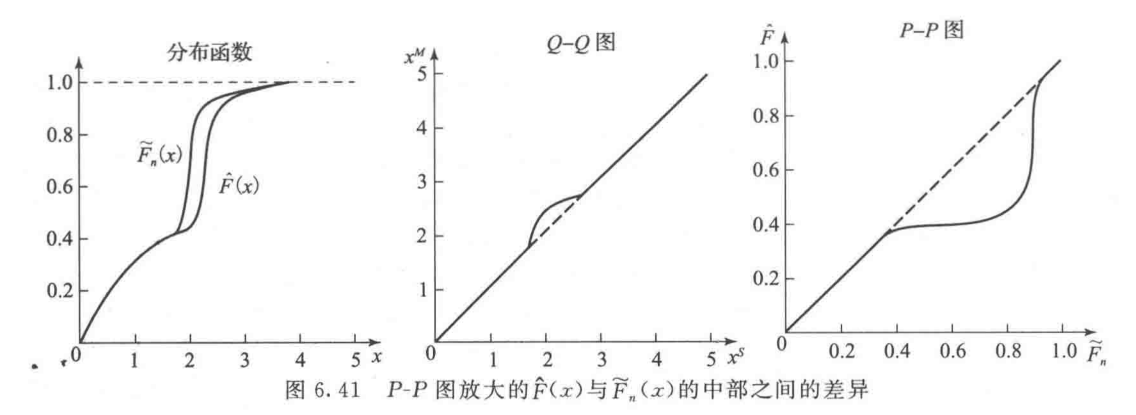


图 6.41 P-P 图放大的$\hat{F}(x)$与$\widetilde{F}_n(x)$的中部之间的差异

上述的 Q-Q 图和 P-P 图的公式隐含假设了 X_i 不同(没有相同的值)；这肯定不一定总是如此。当 X_i 之间可能不互异时，为了修改定义，令 Y_1，Y_2，…，Y_l 为样本 X_1，X_2，…，X_n 中的不同值，以增序排列，其中 $l \leqslant n$(如果 X_i 不同，那么 $Y_i = X_{(i)}$)($i=1, 2, \cdots, n$)。令 q'_i 定义为：

$$q'_i = (X_j \leqslant Y_i \text{ 的比例}) - \frac{0.5}{n}$$

换言之，$q'_i = \widetilde{F}_n(Y_i)$。然后在 Q-Q 图和 P-P 图中用 q'_i 代替 q_i，用 Y_i 代替 $X_{(i)}$。

Q-Q 图的构建需要计算模型的分位数$\hat{F}^{-1}(q_i)$。这对于均匀分布、指数分布、韦布尔分布和对数逻辑斯蒂分布来说是没有问题的，因为$\hat{F}^{-1}$有闭合形式的表达式可用。对于其他的连续分布，在表 6.11 中要么给出了解决该问题的变换，要么给出了$\hat{F}^{-1}$的数值近似的参考值。在表 6.11 中也给出了相似的方法以计算模型概率$\hat{F}(X_{(i)})$，这是 P-P 图所需要的。在 IMSL 统计库[Visual Numerics, Inc. (2004)]中也有计算$\hat{F}$和$\hat{F}^{-1}$的函数可用，并且 ExpertFit 统计包(参见 6.7 节)可自动完成 Q-Q 图和 P-P 图。

表 6.11 某些数学上难以处理的分布计算$\hat{F}$和$\hat{F}^{-1}$的方法

	$\hat{F}$	$\hat{F}^{-1}$
伽马分布	参见 Bhattacharjee(1970)	参见 Best 与 Roberts(1975)
正态分布	参见 Milton 与 Hotchkiss(1969)	参见 Moro(1995)
对数正态分布	对 $i=1, 2, \cdots, n$，$Y_i=\ln X_i$，拟合一个正态分布；参见第 6.2.2 小节	与$\hat{F}$相同
贝塔分布	参见 Bosten 与 Battiste(1974)	参见 Cran，Martin，与 Thomas(1977)
皮尔逊 V 型分布	对 $i=1, 2, \cdots, n$，$Y_i=1/X_i$，拟合一个伽马分布；参见第 6.2.2 小节	与$\hat{F}$相同
皮尔逊Ⅵ型分布	对 $i=1, 2, \cdots, n$，$Y_i=X_i/(1+X_i)$，拟合一个贝塔分布；参见第 6.2.2 小节	与$\hat{F}$相同
约翰逊 S_B分布	参见正态分布	与$\hat{F}$相同
约翰逊 S_U分布	参见正态分布	与$\hat{F}$相同

例 6.13 图 6.42 给出了拟合的指数分布和到达间隔时间数据的 Q-Q 图。除了较大的 q'_i 值外，该图是相当线性的。这并不罕见，因为 Q-Q 图在$\hat{F}(x)$和$\widetilde{F}_n(x)$都接近 1 的时候会放大它们之间的小差异。相应的 P-P 图如图 6.43所示。它的线性特征表明拟合的指数分布的中部与真实的基本分布的中部非常吻合。 ◀

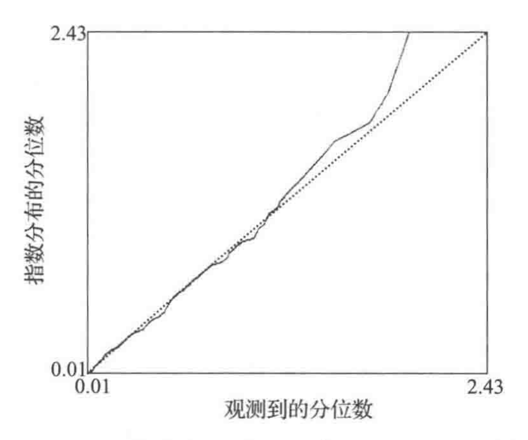


图 6.42　指数分布和到达间隔时间数据的 Q-Q 图

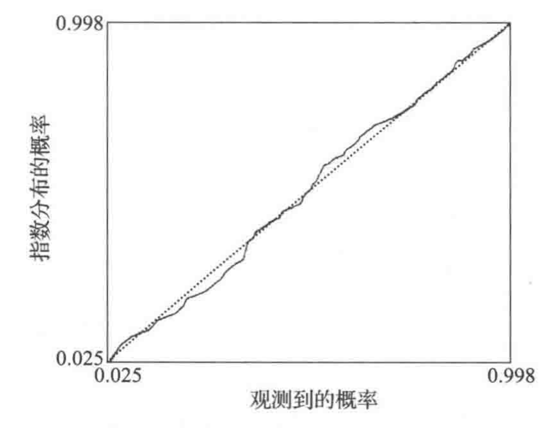


图 6.43　指数分布和到达间隔时间数据的 P-P 图

例 6.14 拟合的几何分布和需求量数据的 P-P 图如图 6.44 所示。我们再次发现 P-P 图是相当线性的，表明几何分布和真实分布是吻合的。◀

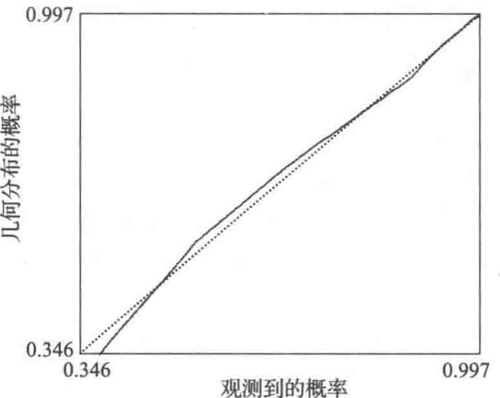


图 6.44　几何分布和需求量数据的 P-P 图

6.6.2　拟合优良度检验

拟合优良度检验是一个统计假设检验(参见第 4.5 节)，用于正式评估观测 X_1，X_2，…，X_n 是否对具有分布函数$\hat{F}$的某特定分布的独立采样。也就是说，拟合优良度检验可以用于检验下面的原假设：

H_0：X_i 是独立同分布的随机变量，具有分布函数$\hat{F}$。

在讨论几种具体的拟合优良度检验之前，需要对这些检验的正式结构和性质进行说明。首先，未能拒绝 H_0 并不意味着"接受 H_0 为真"。这些检验对于中小规模样本量 n 一般不是很有效；也就是说，它们对于数据和拟合分布之间微小的差异不敏感。相反，它们应当看做公正地检测总差别的系统方法。另一方面，如果 n 很大，那么这些检验将几乎永远拒绝 H_0[参见 Gibbons(1985，第 76 页)]。因为 H_0 实际上永远不可能精确地为真，当 n 很大时即使对假设分布很小的偏离也会被检测到。这是这些检验不好的地方，因为一般分布"接近"正确就足够了。

χ^2 检验

最古老的拟合优良度检验是 χ^2 检验，时间至少可以追溯到 K. Pearson(1900)年的文章。正如下面要讲到的，χ^2 检验可以看做直方图与拟合的密度或质量函数的更加正式的对比(参见第 6.6.1 小节中的频率比较)。

为了在连续或离散情况下计算 χ^2 检验统计量，必须首先将拟合分布的整个范围划分为 k 个相邻的区间$[a_0, a_1)$，$[a_1, a_2)$，…，$[a_{k-1}, a_k)$，其中，可能 $a_0=-\infty$，而对应的第一个区间为$(-\infty, a_1)$，或者 $a_k=+\infty$，或者两个都有。然后，对于 $j=1, 2, \cdots, k$，我们统计：

$$N_j = \text{第 } j \text{ 个区间}[a_{j-1}, a_j) \text{ 上 } X_i \text{ 的个数}$$

注意 $\sum_{j=1}^{k} N_j = n$。下面，如果从拟合分布中采样，计算期望落在第 j 个区间上 X_i 的期望比例 p_j。在连续情况下，有：

$$p_j = \int_{a_{j-1}}^{a_j} \hat{f}(x)\mathrm{d}x$$

其中，$\hat{f}$为拟合分布的密度。

对于离散数据，有：

$$p_j = \sum_{a_{j-1} \leqslant x_i \leqslant a_j} \hat{p}(x_i)$$

其中，$\hat{p}$为拟合分布的质量函数。

最后，检验统计为：

$$\chi^2 = \sum_{j=1}^{k} \frac{(N_j - np_j)^2}{np_j}$$

由于 np_j 是 H_0 为真时 nX_i 落在第 j 个区间上的期望个数(参见习题 6.17)，如果拟合很好，我们期望 χ^2 会很小。因此，如果 χ^2 太大，拒绝 H_0。检验的精确形式依赖于是否根据数据对拟合分布的任何参数进行了估计。

首先，假设拟合分布的所有参数都已知，也就是，不以任何方式使用数据来指定拟合分布[这种所有参数都已知的情况在实际中可能很少出现，但是在仿真中它至少有两种应用：①在泊松过程检验中(本节后面)，我们要检验到达时间是否可以看做是独立同分布 $U(0, T)$随机变量，其中，T 为与数据独立的常数；以及②在实验检验随机数发生器(第 7.4.1 小节)时，我们检验 $U(0, 1)$分布]。那么，如果 H_0 为真，那么 χ^2 以分布(当 $n \to +\infty$时)会收敛成自由度为 $k-1$ 的 χ^2 分布，也就是 $\Gamma[(k-1/2), 2]$分布。因此对于大的 n，如果 $\chi^2 > \chi^2_{k-1,1-\alpha}$，检验以近似水平 α 通过拒绝 H_0 得到(参见图6.45)，其中 $\chi^2_{k-1,1-\alpha}$为自由度为 $k-1$ 的 χ^2 分布的上 $1-\alpha$ 临界点($\chi^2_{k-1,1-\alpha}$的值可以在本书结尾附录中表 A.2 中找到)。注意 χ^2 检验只是随着 $n \to +\infty$渐近以水平 α 有效。

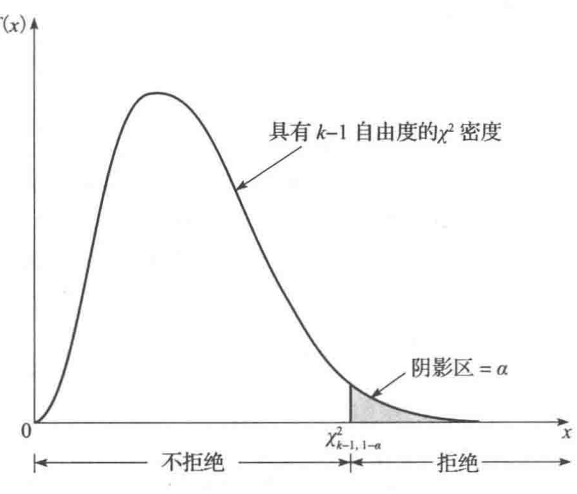


图 6.45 当所有参数都已知时 χ^2 分布检验

其次，假设为了指定拟合分布，必须从数据中估计 $m(m \geqslant 1)$个参数。当使用 MLE 时，Chernoff 和 Lehmann (1954)证明过，如果 H_0 为真，那么随着 $n \to +\infty$，χ^2 的分布函数收敛位置介于自由度为 $k-1$ 和自由度为 $k-m-1$ 的 χ^2 分布之间(参见图 6.46，其中 F_{k-1} 和 F_{k-m-1}分别代表了自由度为 $k-1$ 和自由度为 $k-m-1$ 的 χ^2 分布，虚线分布函数为 $n \to +\infty$时 χ^2 的分布函数的收敛结果)。如果我们令 $\chi^2_{1-\alpha}$为 χ^2 的渐近分布的上 $1-\alpha$ 临界点，那么

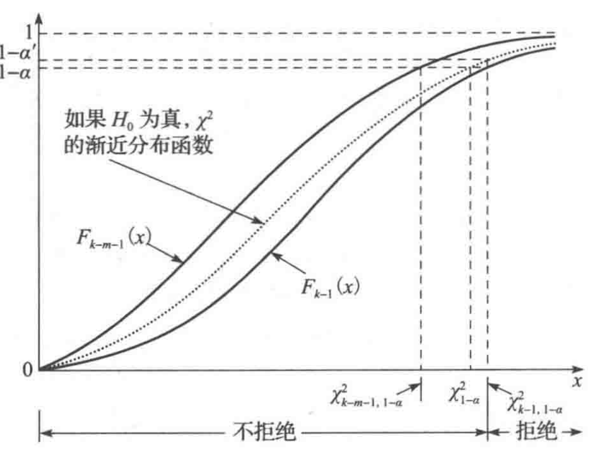


图 6.46 当 m 个参数都通过 MLE 得到时的 x^2 检验

$$\chi^2_{k-m-1,1-\alpha} \leqslant \chi^2_{1-\alpha} \leqslant \chi^2_{k-1,1-\alpha}$$

如图 6.46 所示，不幸的是，$\chi^2_{1-\alpha}$一般不知道。显然，如果 $\chi^2 > \chi^2_{k-1,1-\alpha}$，应当拒绝 H_0，而如果 $\chi^2 < \chi^2_{k-1,1-\alpha}$，不应当拒绝 H_0；当

$$\chi^2_{k-m-1,1-\alpha} \leqslant \chi^2 \leqslant \chi^2_{k-1,1-\alpha}$$

时情况比较含糊。往往建议只有当 $\chi^2 > \chi^2_{k-1,1-\alpha}$时，才拒绝 H_0，因为这样比较保守；也就是说，犯类型Ⅰ错误的实际概率 α'[H_0 为真的时候拒绝 H_0(参见第 4.5 节)]至少与声称概率 α 一样小(参见图 6.46)。然而，这种选择会造成检验能力的损失(拒绝假 H_0 的概率)。通常，m 都不超过 2，而且如果 k 相当大的话，$\chi^2_{k-m-1,1-\alpha}$和 $\chi^2_{k-1,1-\alpha}$之间的差别也不会太大，因此，在所有参数未知的情形下，当(且仅当)$\chi^2 > \chi^2_{k-1,1-\alpha}$，拒绝 H_0。在图 6.46 中给出了 χ^2 的拒绝域。

执行 χ^2 检验最麻烦的事情就是选择间隔的数目和大小。这是个难题，还没有明确的方法能够保证对所有的假设分布以及所有样本大小得到较好的结果，具有有效性(实际检验水平接近希望水平 α)，以及高效率。然而有为数不多的可遵循的指导性意见。首先，如果区间选择都满足 $p_1 = p_2 = \cdots = p_k$，这称为等概率方法，那么区间选择的不确定性就可以消除。在连续情况下，因为拟合分布的分布函数必须求逆，这样做也许不方便(参见下面例 6.15)。此外，对于离散分布，一般只可以使 p_j 近似相等(参见例 6.16)。

现在讨论如何选择区间，以确保检验“有效”。令 $a = \min\limits_{1 \leqslant j \leqslant k} np_j$，$y(5)$为 np_j 小于 5 的个数。基于广泛的理论和实验研究(对于所有参数都已知的情况)，Yarnold(1970)认为 χ^2 检验在 $k \geqslant 3$ 和 $a \geqslant 5y(5)/k$ 时近似有效。对于等概率区间，对于所有 j，如果 $k \geqslant 3$ 和 $np_j \geqslant 5$，这两个条件均会满足。

现在我们将注意力转到 χ^2 检验的能力上。一个检验如果对于 H_0 为假的时候比 H_0 为真的时候更加倾向于拒绝 H_0，那么就称该检验就是无偏的；或者，换言之，能力大于类型Ⅰ错误的概率。无此性质的检验肯定不是理想的。可以证明 χ^2 检验对于等概率方法总是无偏的[参见 Kendall 和 Stuart(1979，第 455－461 页)]。如果 np_j 之间不相等(且很多是小的)，有可能得到一个高偏的有效检验[参见 Haberman(1988)]。

一般说来，尚无挑选区间的规则以保证对于所有备选分布都得到很高的能力。对于一个特定的原分布，固定的样本大小 n，以及等概率方法，Kallengerg、Oosterhoff 和 Schriever(1985)实验表明，对于某些备选分布来说，能力是区间个数 k 的增函数，而对于其他备选分布来说，能力则是区间个数 k 的减函数。令人惊奇的是，他们还发现在某些情况下当尾部 np_j 较小时能力较大(参见习题 6.18)。

在挑选区间缺少明确的方法的情况下，我们推荐使用等概率方法，并且在连续情况下对于所有的 j，使得 $np_j \geqslant 5$。这能保证检验有效、无偏。在离散情况下，我们建议使 np_j 接近相等，并且所有 np_i 至少是 5。挑选区间缺少明确的方法是 χ^2 检验的主要缺点。在某些情况下，依区间的确定不同，同样的数据集可能会导致完全不同的结论，如例 6.17 所示。然而 χ^2 检验仍然得到广泛应用，因为它可以用于任何假设的分布；正如下面将要看到的那样，其他的拟合优良度检验没有这样广泛的适用性。

例 6.15 下面应用 χ^2 检验方法将表 6.7 中 $n=219$ 个到达时间间隔数据，与分布函数为$\hat{F}(x) = 1 - e^{-x/0.399}$，$x \geqslant 0$ 的拟合的指数分布作比较。如果构造，比如说，20 个区间，$k=20$，$p_j = 1/k = 0.05$，$j=1, 2, \cdots, 20$，那么 $np_j = 219 \times 0.05 = 10.950$，因此它满足了方法中对于区间的选择满足 p_j 等大小，以及 $np_j \geqslant 5$ 的要求。在这种情况下，很容易找到 a_j，因为能对$\hat{F}$求逆。因此，令 $a_0 = 0$，且 $a_{20} = +\infty$，以及对于 $j=1, 2, \cdots, 19$，我们要使 a_j 满足$\hat{F}(a_j) = j/20$；这等同于令 $a_j = -0.399\ln(1-j/20)$，$j=1, 2, \cdots, 19$，因

为 $a_j=\hat{F}^{-1}(j/20)$(对于连续分布，如正态分布、伽马分布、贝塔分布，分布函数反函数没有简单的闭合形式。然而在这种情况下，F^{-1} 的值可以用数值方法得到；参考表 6.11)。表 6.12 给出了该检验的计算过程，检验统计的值为 $\chi^2=22.188$。参考本书末尾的附录中表 A.2，看到 $\chi^2_{19,0.90}=27.204$，χ^2 没有超过它，因此在 $\alpha=0.1$ 的水平上不拒绝 H_0(注意，对于某个大的水平值如 $\alpha=0.25$，也不会拒绝 H_0)。因此，这个检验给出了没有理由得出 expo(0.399)分布拟合数据很差的结论。

表 6.12 到达间隔时间数据的 χ^2 拟合优良度检验

j	区间	N_j	np_j	$\frac{(N_j-np_j)^2}{np_j}$
1	[0, 0.020)	8	10.950	0.795
2	[0.020, 0.042)	11	10.950	0.000
3	[0.042, 0.065)	14	10.950	0.850
4	[0.065, 0.089)	14	10.950	0.850
5	[0.089, 0.115)	16	10.950	2.329
6	[0.115, 0.142)	10	10.950	0.082
7	[0.142, 0.172)	7	10.950	1.425
8	[0.172, 0.204)	5	10.950	3.233
9	[0.204, 0.239)	13	10.950	0.384
10	[0.239, 0.277)	12	10.950	0.101
11	[0.277, 0.319)	7	10.950	1.425
12	[0.319, 0.366)	7	10.950	1.425
13	[0.366, 0.419)	12	10.950	0.101
14	[0.419, 0.480)	10	10.950	0.082
15	[0.480, 0.553)	20	10.950	7.480
16	[0.553, 0.642)	9	10.950	0.347
17	[0.642, 0.757)	11	10.950	0.000
18	[0.757, 0.919)	9	10.950	0.347
19	[0.919, 1.195)	14	10.950	0.850
20	[1.195, +∞)	10	10.950	0.082
				$\chi^2=22.188$

◀

例 6.16 作为离散情况下 χ^2 拟合优良度检验说明，检验几何分布 geom(0.346) 对表 6.9 中需求量数据的拟合的吻合程度如何。同通常的离散分布情形一样，不能使 p_j 完全相等，但是将所定义的质量函数 $\hat{p}(x)$(这里是非负整数)上相邻点划分成组，就可以定义区间使得 p_j 大致相等。做到这一点的一种方法是，注意到拟合分布的众数是 0，即 $\hat{p}(0)=0.346$ 为质量函数的最大值。大的众数值限制了区间选择，因此，只能划分 3 个区间，表 6.13 也给出了 χ^2 检验的计算。特别是，$\chi^2=1.930$，它比临界值 $\chi^2_{2,0.90}=4.605$ 要小。因此，在 $\alpha=0.1$ 的水平上不会拒绝 H_0，也没有理由相信用 geom(0.346)分布来拟合需求量数据是不好的。

表 6.13 需求量数据的 χ^2 拟合优良度检验

j	区间	N_j	np_j	$\frac{(N_j-np_j)^2}{np_j}$
1	{0}	59	53.960	0.471
2	{1, 2},	50	58.382	1.203

（续）

j	区间	N_j	np_j	$\frac{(N_j-np_j)^2}{np_j}$
3	{3, 4, …},	47	43.658	0.256
				$\chi^2=1.930$

◀

例 6.17 如果对图 6.3 所示的 856 个装载时间来拟合对数逻辑斯蒂分布，那么，得到的形状和比例参数的 MLE 分别为$\hat{\alpha}=8.841$，$\hat{\beta}=0.822$。使用 $k=10$，20 和 40 等概率区间执行 $\alpha=0.1$ 的 χ^2 检验，结果如表 6.14 所示(注意，k 的所有三个选择都满足 $np_j \geqslant 5$ 的建议)。我们看到对数逻辑斯蒂分布在 20 个区间的情况下被拒绝，而在 10 个和 40 个区间的情况下没有被拒绝。

表 6.14 装载时间 χ^2 拟合优良度检验

k	统计量	临界值	检验结果
10	11.383	14.684	不拒绝
20	27.645	27.204	拒绝
40	50.542	50.660	不拒绝

◀

科尔莫戈罗夫-斯米尔洛夫检验

正如我们刚刚看到的一样，χ^2 检验可以看做将数据直方图与拟合分布的密度函数或者质量函数进行比较正式的对比。在连续情况下应用 χ^2 检验的一个实际困难，即如何确定区间。另一方面，拟合优良度的科尔莫戈罗夫-斯米尔洛夫(Kolmogorov-Smirňor，K-S)检验将经验分布函数与假设分布的分布函数$\hat{F}$作对比。正如我们将要看到的一样，K-S 检验不需要以任何形式对数据分组，因此没有任何信息丢失；这些解决了区间划分这个麻烦问题。K-S 检验另一个优点就是，它们对于任意样本大小 n(在所有参数都已知的情况下)都是(严格)有效的，而 χ^2 检验只在渐近意义上有效。最后，在很多可选分布的情况下，K-S 检验能力比 χ^2 检验强；参见，例如，Stephens(1974)。

然而，至少目前 K-S 检验还是有一些缺点。最严重的是，它的适用范围比 χ^2 检验有更多的限制。首先，对于离散数据，需要的临界值不易得到，必须通过计算一组复杂的方程[参见 Conover(1999，第 435-437 页)，Gleser(1985)，以及 Pettitt 和 Stephens(1977)]。其次，只有当假设分布的所有参数已知，且分布为连续时，K-S 的原始形式才有效；即参数不能由数据来估计。然而，K-S 检验已经有很多发展，容许用于正态(对数正态)分布、指数分布、韦布尔分布和对数逻辑斯蒂分布的情况下的参数估计。虽然原始形式(所有参数已知)中的 K-S 检验经常直接用于带有估计参数的任意连续分布和离散分布，实际上，这会导致保守检验[参见 Conover(1999，第 432，442 页)]。也就是说，类型 I 错误的概率比给定的小，带来相应的能力损失。

为了定义 K-S 统计量，将使用由式(6.5)定义的经验分布函数 $F_n(x)$，它是一个(右连续)阶跃函数，这样，对于 $i=1, 2, \cdots, n$，$F_n(X_{(i)})=i/n$(参见习题 6.19)。如果$\hat{F}(x)$为拟合的分布函数，一个很自然的评估拟合优良度方法就是对函数 F_n 和$\hat{F}$之间接近程度进行某一类测量。K-S 检验统计量 D_n 就是对于 x 的所有值，F_n 和$\hat{F}$之间的最大(垂向)距离，其形式化的定义是：

$$D_n = \sup_x \{ |F_n(x) - \hat{F}(x)| \}$$

数的集合 A 的“sup”为大于或等于 A 的所有数的最小值。这里使用“sup”而不是

更为熟悉的“max”是因为，在某些情况下，最大值可能不好定义。例如，如果 $A=(0,1)$，那么没有最大，但是“sup”为 1。统计量 D_n 为：

$$D_n^+ = \max_{1\leqslant i\leqslant n}\left\{\frac{i}{n}-\hat{F}(X_{(i)})\right\},\quad D_n^- = \max_{1\leqslant i\leqslant n}\left\{\hat{F}(X_{(i)})-\frac{i-1}{n}\right\}$$

并且最后令

$$D_n = \max\{D_n^+, D_n^-\}$$

图 6.47 给出了一个 $n=4$ 的例子，其中，$D_n=D_n^+$[注意！给出的计算 D_n 的公式往往不对。特别是，有时人们看到将：

$$D_n' = \max_{1\leqslant i\leqslant n}\left\{\left|\frac{i}{n}-\hat{F}(X_{(i)})\right|\right\}$$

作为 D_n 的“公式”。对于图 6.47 所示的情形，确实是 $D_n'=D_n$。然而考虑图 6.48 所示情况，这里 $D_n'=\hat{F}(X_2)-\frac{2}{4}$。但是 D_n 的正确值是 $\hat{F}(X_2)-\frac{1}{4}$，它正好发生在 $x=X_{(2)}$ 前。显然，在这种情形下 $D_n'\neq D_n$]。直接计算 D_n^+，D_n^- 需要对数据排序以得到 $X_{(i)}$。然而，Gonzalez，Sahni 和 Franta(1977)提供了无排序计算 D_n^+ 和 D_n^- 的方法。

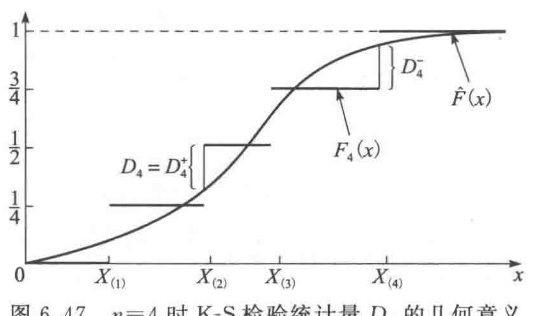


图 6.47 $n=4$ 时 K-S 检验统计量 D_n 的几何意义

图 6.48 K-S 检验统计量 D_n 不等于 D_n' 的例子

显然，D_n 较大表明拟合效果很差，从而检验的形式就是，如果 D_n 超过某个常数 $d_{n,1-\alpha}$，则拒绝原假设 H_0，其中，α 为规定的检验水平。临界点 $d_{n,1-\alpha}$的数值依赖于如何给定所假设的分布，因而我们区分以下几种情况。

情况 1

假设(自然)$\hat{F}$连续，如果$\hat{F}$所有参数都已知，即无$\hat{F}$的参数以任何形式由数据估计得到的，D_n 的分布不依赖于$\hat{F}$。这个相当重要的事实意味着，对于所有连续分布形式，$d_{n,1-\alpha}$的值用一张表就足够了；这些表在很多地方都有[参见，例如。Owen(1962)]。Stephens(1974)发明了一个精确的近似式，除了一个小表外去掉了所有要求；不检验 $D_n>d_{n,1-\alpha}$，如果：

$$\left(\sqrt{n}+0.12+\frac{0.11}{\sqrt{n}}\right)D_n > c_{1-\alpha}$$

拒绝 H_0，其中，$c_{1-\alpha}$的值(它不依赖于 n)在表 6.15 中所有参数已知行中给出。这种所有参数已知的例子就是 K-S 检验的原始形式。

表 6.15 调整 K-S 检验统计的修改的临界值 $c_{1-\alpha}$，$c'_{1-\alpha}$，$c''_{1-\alpha}$，

情形	调整的检验统计量	$1-\alpha$				
		0.850	0.900	0.950	0.975	0.990
所有参数已知	$\left(\sqrt{n}+0.12+\frac{0.11}{\sqrt{n}}\right)D_n$	1.138	1.224	1.358	1.480	1.628
$N(\overline{X}(n),\ S^2(n))$	$\left(\sqrt{n}-0.01+\frac{0.85}{\sqrt{n}}\right)D_n$	0.775	0.819	0.895	0.955	1.035
$\mathrm{expo}(\overline{X}(n))$	$\left(D_n-\frac{0.2}{n}\right)\left(\sqrt{n}+0.26+\frac{0.5}{\sqrt{n}}\right)$	0.926	0.990	1.094	1.190	1.308

情况 2

假定所假设的分布为 $N(\mu, \sigma^2)$，其中，μ 和 σ^2 都未知。通过 $\overline{X}(n)$ 和 $S^2(n)$ 分别估计 μ 和 σ^2，并将分布函数 $\hat{F}$ 定义为 $N(\overline{X}(n), S^2(n))$ 分布；即令 $\hat{F}(x)=\Phi\{[x-\overline{X}(n)]/\sqrt{S^2(n)}\}$，其中，$\Phi$ 为标准正态分布的分布函数。使用这个 $\hat{F}$（它有估计的参数），使用相同的方法计算 D_n，但是必须用不同的临界点。Lilliefors(1967)（通过蒙特卡罗仿真）估计出临界点 D_n，它是 n 和 $1-\alpha$ 的函数。Stephens(1974)执行了更多的蒙特卡罗仿真，并提供了更加精确的近似式，不需要大表格；即，如果：

$$\left(\sqrt{n}-0.01+\frac{0.85}{\sqrt{n}}\right)D_n > c'_{1-\alpha}$$

拒绝 H_0，其中，$c'_{1-\alpha}$ 的值在表 6.15 的 $N(\overline{X}(n), S^2(n))$ 行中给出（这种情形包含了对数正态分布的 K-S 检验，如果我们假设分布为对数正态分布，X_i 是基础数据点的对数的话，参见第 6.2.2 小节）。

情况 3

假定假设的分布为 expo(β)，β 为未知。现在，通过其 MLE 的 $\overline{X}(n)$ 来估计 β，并且定义 $\hat{F}$ 为 expo($\overline{X}(n)$)分布函数；也就是 $\hat{F}=1-e^{-x/\overline{X}(n)}$，$x\geqslant 0$。在这种情况下，临界点 D_n 最初是由 Lilliefors(1969)在蒙特卡罗研究中估计出来的，而精确的表是后来由 Durbin(1975)得到的[也可参见 Margolin 和 Maurer(1976)]。这种情况下的 Stephens(1974)近似式为：

$$\left(D_n-\frac{0.2}{n}\right)\left(\sqrt{n}+0.26+\frac{0.5}{\sqrt{n}}\right) > c''_{1-\alpha}$$

如果满足，则拒绝 H_0。其中，$c''_{1-\alpha}$ 可以在表 6.15 的 expo($\overline{X}(n)$)行中找到。

情况 4

假定假设的分布为韦布尔分布，形状参数 α 与比例参数 β 都未知。通过它们对应的 MLE 的 $\hat{\alpha}$ 和 $\hat{\beta}$（参见第 6.2.2 小节中韦布尔分布的 MLE 的讨论）来估计这些参数。而 $\hat{F}$ 取自 $W(\hat{\alpha}, \hat{\beta})$ 分布函数 $\hat{F}(x)=1-\exp[-(x/\hat{\beta})^{\hat{\alpha}}]$，$x\geqslant 0$，且 D_n 也通常的方式计算得到。那么如果调整的 K-S 统计量 $\sqrt{n}D_n$ 大于表 6.16 中修正的临界值 $c^*_{1-\alpha}$[参见 Chandra, Singpurwalla 和 Stephens(1981)]，则拒绝 H_0。注意，只有对一定的样本量 n 才有临界值，而且，幸运的是，$n=50$ 以及 $+\infty$（极大的样本量）的临界值非常相似[Littell、McClave 和 Offen(1979)给出了 n 小于 50 的临界值]。

表 6.16　韦布尔分布 K-S 检验的修正临界值 $c^*_{1-\alpha}$

n	$1-\alpha$			
	0.900	0.950	0.975	0.990
10	0.760	0.819	0.880	0.944
20	0.779	0.843	0.907	0.973
50	0.790	0.856	0.922	0.988
∞	0.803	0.874	0.939	1.007

情况 5

假定假设的分布为对数逻辑斯蒂分布，形状参数 α、比例参数 β 都未知。这里，令 X_i 为基础数据点的对数。基于 X_i 通过各自 MLE 的 $\hat{\alpha}$ 和 $\hat{\beta}$ 估计这些参数（参见第 6.2.2 小节）。$\hat{F}(x)$ 也取对数分布函数：

$$\hat{F}(x)=(1+e^{[-(x-\ln\hat{\beta})]\hat{\alpha}})^{-1}, \quad -\infty < x < +\infty$$

并且 D_n 以通常的方式计算得到。那么如果调整的 K-S 统计量 $\sqrt{n}D_n$ 大于表 6.17 中修正的临界值 $c^+_{1-\alpha}$[参见 Stephens(1979)]，则拒绝 H_0。注意，只有对于一定的样本量，n 才有临

界值，而且，幸运的是，$n=50$ 以及 $+\infty$ 的临界值非常相似。

表 6.17 对数逻辑斯蒂分布 K-S 检验的修正临界值 $c_{1-\alpha}^{+}$

n	$1-\alpha$			
	0.900	0.950	0.975	0.990
10	0.679	0.730	0.774	0.823
20	0.698	0.755	0.800	0.854
50	0.708	0.770	0.817	0.873
∞	0.715	0.780	0.827	0.886

例 6.18 在例 6.15 中，对于表 6.7 中的到达间隔时间数据，使用 χ^2 检验来检查拟合分布 expo(0.399)的拟合优度。也可以使用上面的情况 3 对 $\hat{F}(x)=1-e^{-x/0.399}$，$x\geqslant 0$ 应用 K-S 检验。应用 D_{219}^{+} 和 D_{219}^{-} 的公式，我们得到 $D_{219}=0.047$，因此调整的检验统计量为：

$$\left(D_{219}-\frac{0.2}{219}\right)\left(\sqrt{219}+0.26+\frac{0.5}{\sqrt{219}}\right)=0.696$$

由于 0.696 小于 $0.990=c''_{0.90}$（表 6.15 最后一行），在 $\alpha=0.10$ 的水平上我们不拒绝 H_0。◀

安德森-达林检验⊖

K-S 检验的一个可能的缺点就是，对于每个 x 的值，给差值 $|F_n(x)-\hat{F}(x)|$ 以相同的权重，然而很多感兴趣的分布主要是在尾部不同。另一方面，安德森-达林（Anderson-Darling）检验（A-D 检验）[参见 Anderson 和 Darling(1954)]设计成检测尾部的差异，从而在面对多种可选分布时比 K-S 检验有更强的能力。A-D 检验统计量 A_n^2 定义为：

$$A_n^2=n\int_{-\infty}^{+\infty}[F_n(x)-\hat{F}(x)]^2\psi(x)\hat{f}(x)\mathrm{d}x$$

其中，权重函数 $\psi(x)=1/\{\hat{F}(x)[1-\hat{F}(x)]\}$。因此，$A_n^2$ 就是差值平方 $[F_n(x)-\hat{F}(x)]^2$ 的加权平均值，$\hat{F}(x)$ 接近 1（右尾）和接近 0 时（左尾）权重最大。如果令 $Z_i=\hat{F}(x_{(i)})$，$i=1,2,\cdots,n$，那么，可以证明

$$A_n^2=\left(-\left\{\sum_{i=1}^{n}(2i-1)[\ln Z_i+\ln(1-Z_{n+1-i})]\right\}/n\right)-n$$

这也是实际计算中所使用的统计量的形式。由于 A_n^2 是一个“加权距离”，如果 A_n^2 超过某个临界值 $a_{n,1-\alpha}$，检验的形式就是拒绝原假设 H_0，其中，α 为检验的水平。

与 K-S 检验相同的五种连续分布有 A-D 检验的临界值 $a_{n,1-\alpha}$ 可用[参见 Stephens(1974，1976，1977，1979)以及 D'Agostino 和 Stephens(1986，第 134 页)，对于离散情形的讨论参见 Gleser(1985)]。此外，$\hat{F}(x)$ 的计算方法与前面相同；参见下面例 6.19。A-D 检验的性能可以通过使用调整检验统计量（所有参数都已知的情况除外）以及修正临界值得到提高，这在表 6.18 中给出。如果调整的检验统计量大于修正的临界值，那么拒绝 H_0。

表 6.18 调整的 A-D 检验统计量的修正临界值

情形	调整的检验统计量	$1-\alpha$			
		0.900	0.950	0.975	0.990
所有参数已知	$A_n^2(n\geqslant 5)$	1.933	2.492	3.070	3.857
$N(\overline{X}(n),S^2(n))$	$\left(1+\frac{4}{n}-\frac{25}{n^2}\right)A_n^2$	0.632	0.751	0.870	1.029

⊖ 第一次阅读可跳过本节。

（续）

情形	调整的检验统计量	$1-\alpha$			
		0.900	0.950	0.975	0.990
$\text{expo}(\overline{X}(n))$	$\left(1+\frac{6}{n}\right)A_n^2$	1.062	1.321	1.591	1.959
$W(\hat{\alpha},\ \hat{\beta})$	$\left(1+\frac{0.2}{\sqrt{n}}\right)A_n^2$	0.637	0.757	0.877	1.038
$LL(\hat{\alpha},\ \hat{\beta})$	$\left(1+\frac{0.25}{\sqrt{n}}\right)A_n^2$	0.563	0.660	0.769	0.906

D'Agostino 和 Stephens(1986，第 151-156 页)给出了一个执行伽马分布的 A-D 检验的步骤，其中临界值在通过表中插值得到。A-D 检验也可以用于皮尔逊类型V分布检验，它利用这样一个事实，若 X 为皮尔逊类型V分布，那么 $1/X$ 为伽马分布(参见第 6.2.2 小节)。

例 6.19 使用 A-D 检验的情形 3 来观察拟合的指数分布 $\hat{F}(x)=1-e^{-x/0.399}$ 是否在水平 $\alpha=0.10$ 上对到达间隔时间数据提供了一个好模型。求得 $A_{219}^2=0.558$，从而，调整的检验统计量为：

$$\left(1+\frac{0.6}{219}\right)A_{219}^2=0.560$$

由于 0.560 比修正的临界值 1.062(表 6.18 第 3 行)小，在水平 $\alpha=0.10$ 上不拒绝 H_0。◀

泊松过程检验[⊖]

假设对一个固定的时间区间[0，T]观察一个泊松(Poisson)过程(参见第 6.12.1 小节)，其中，T 为常数，在开始观察前确定。令 n 为在[0，T]区间内观察到的事件数目；对于 $i=1，2，\cdots，n$，令 t_i 为第 i 个事件发生的时间{因此，$0\leqslant t_1\leqslant t_2\leqslant\cdots\leqslant t_n\leqslant T$。如果 $t_n<T$，那么在区间(t_n，T]内没有事情发生}。那么 t_1，t_2，…，t_n 的联合分布以下面的方式同 $U(0，T)$关联。假设 Y_1，Y_2，…，Y_n(n 同前面一样)为独立同分布随机变量，分布为 $U(0，T)$，且令 $Y_{(1)}$，$Y_{(2)}$，…，$Y_{(n)}$ 为它们对应的有序统计量(参见第 6.2.4 小节)。泊松过程的一个性质就是 t_1，t_2，…，t_n 的联合分布与 $Y_{(1)}$，$Y_{(2)}$，…，$Y_{(n)}$ 相同[证明参见 Ross(2003，第 303 页)]。

解释这个性质的一个方法就是，如果有人只是简单的告诉我们 t_1，t_2，…，t_n 值，而不告诉我们 t_i 的获得是作为某个事件序列第 i 个事件的时间的话，那么它们看起来(在统计意义上)就像是这 n 个数字是来自 $U(0，T)$分布 n 个独立同分布随机变量值的样本，然后将它们排成增序。另一种方法，可以将这个性质表述为，如果我们将 t_1，t_2，…，t_n 看做是未排序的随机变量，那么它们都是 $U(0，T)$独立同分布随机变量。这也是为什么我们有时将所描述的泊松过程看做事件“随机”地发生的过程，因为事件发生的时刻随时间变化是均匀分布的。

在任何情况下，这个性质都提供了一种检验原假设的不同方法，即用泊松过程产生所观察的事件序列(检验这个假设的一种方法，即检验到达间隔时间是否是独立同分布的指数分布随机变量；参见第 6.12.1 小节和例 6.15，例 6.18 及例 6.19)。使用任何合适的检验程序，只需要简单地检验 t_1，t_2，…，t_n 是否是独立同均匀分布 $U(0，T)$随机变量即可。

例 6.20 表 6.7 所示的到达间隔时间数据是在一个固定的 90 分钟周期中收集的，在这个区间共记录了 $n=220$ 个到达(事前就决定观测该过程在正好下午 5：00 开始，而不是

⊖ 第一次阅读可跳过本节。

5：00之后第一个到达发生的时间开始。另外，在下午6：30数据收集即刻结束，而不管后面任何到达的发生。数据收集设计成这种方式，即独立于实际事件时间，对该检验的有效性是很重要的)。到达时间为 $t_1=1.53$，$t_2=1.98$，…，$t_{220}=88.91$(下午5：00之后以分钟为单位)。为了检验这些数字是否可以看做独立的 $U(0, 90)$ 分布，我们使用所有参数已知的 χ^2 检验和K-S检验[拟合分布的密度和分布函数分别为 $\hat{f}(x)=1/90$，$\hat{F}(x)=x/90$，$0\leqslant x\leqslant 90$。还要注意到我们的"数据"点已经很方便地排序了]。我们执行 χ^2 检验，$k=17$ 个等长区间[0，5.294)，[5.294，10.588)，…，[84.706，90]，那么 $np_j=220/17=12.941$，$j=1, 2, \cdots, 17$。得到的 χ^2 值为13.827，且由于 $\chi^2_{16,0.90}=23.542$，我们在水平为0.10上不能拒绝到达发生服从泊松过程这一原假设。K-S检验结果 $D_{220}=0.045$，且由表6.15中所有参数已知行中调整的检验统计量的值为0.673。由于这个远小于 $c_{0.90}=1.224$，再次在水平为0.10上不能拒绝该原假设。◀

6.7 ExpertFit 软件与扩展的例子

在一些情况下，执行本章讨论的统计程序会有困难，费时，且易出错。例如，等区间 χ^2 检验需要分布函数可逆，但某些分布(例如，正态分布和伽马分布)没有闭合形式可用。因此，在这些情况下必须编程，并得到逆分布函数的数值近似。此外，K-S检验在教科书和软件包中也经常被错误地表述或者应用。这些原因导致了ExpertFit分布拟合软件的开发。

商业版本的ExpertFit[参见Averill M. Law & Associates(2013)]会自动精确地确定出40个概率分布中哪个最佳地表示了一个数据集。然后，以合适的格式将所选分布直接输入到大量不同的仿真包中。ExpertFit包含了以下四个模块，它们可以依次用于确定最佳分布。

- "数据"录入、导入数据到ExpertFit中，现实求和统计，并生成直方图；生成相关图与散点图；以及对数据集的均匀性执行克鲁斯卡-沃尔斯(Kruskal-Wallis)检验(参见第6.13节)。
- "模型"使用最大似然法对数据进行分布拟合，根据拟合的质量对分布排序，确定"最佳"分布用于仿真模型时实际上是否足够优良(否则，它建议使用经验分布)。
- "对比"将最佳分布与数据对比，以进一步判断拟合优良度，使用密度直方图、分布函数差异图、概率图、拟合优良度检验等方法。
- "应用"展示并计算所拟合的分布的特征，例如密度函数、矩、概率和分位数；它还将所选分布变成适合所选仿真软件包的格式。

ExpertFit具有以下文档：

- 所有菜单以及结果表格、图像的上下文相关的帮助；
- 在线特征索引，以及教学课程(参见屏幕上方菜单工具栏"帮助"下拉菜单)；
- 包含8个完整的例子的用户指南。

本书配套使用学生版本ExpertFit，同商业版本相比，有以下限制：

- 最大可用样本大小为 $n=100$；
- 可使用连续分布为7种(指数分布、伽马分布、对数逻辑斯蒂分布、对数正态分布、正态分布、均匀分布与韦布尔分布)，离散分布为5种(二项分布、几何分布、负二项分布、泊松分布与均匀分布)；
- 缺少一些高级统计功能；
- 所选分布不能够变成适合所选仿真软件包的格式。

我们下面使用ExpertFit学生版对例6.1(包含在软件中)的 $n=200$ 个服务时间进行一

个全面的分析。在表 6.19 中我们给出服务时间的“数据求和”(即求和统计)，而图 6.49 所示的是对应的直方图，基于区间数 $k=12$，宽度 $\Delta b=0.2$(区间宽度是通过试凑法确定的)。直方图的形状明显地提示基本分布向右偏斜，这排除了正态分布和均匀分布。计算结果支持这一点，$\overline{X}(200)=0.888>0.849=\hat{x}_{0.5}(200)$，$\hat{v}(200)=0.513>0$。此外，$\widehat{cv}(200)=0.515$ 也使真实分布为指数分布的可能性不大，指数分布的方差系数为 1；直方图形状支持这个结论。

表 6.19　服务时间数据统计求和

数据特征	值
源文件	EXAMPLE61
观测数据类型	实型值
观测数据个数	200
最小观测数据的值	0.054
最大观测数据的值	2.131
均值	0.888
中值	0.849
方差	0.210
方差系数	0.515
偏斜度	0.513

“模型”模块中的“自动拟合”选项然后可以用于自动拟合、排序、评估除正态分布之外的连续分布。正态分布对数据不是自动拟合的，因为它可以取负值，这与服务时间的取值范围不一致(如果想做，可以使用“模型”模块的“拟合单个模型”手动设置，正态分布可拟合这些数据)。得到的 ExpertFit 结果屏幕如图 6.50 所示。从“备选模型的相对评价”中可以看到，韦布尔分布排名第一，得到的“相对分数”(见下面)为 100.00，接下来为伽马分布、带估计位置参数的韦布尔分布(附在分布名字的右边标以“E”)，相对分数分别为 83.33 和 62.50。最佳拟合的韦布尔分布的最大似然估计为$\hat{\alpha}=2.036$，$\hat{\beta}=0.943$。

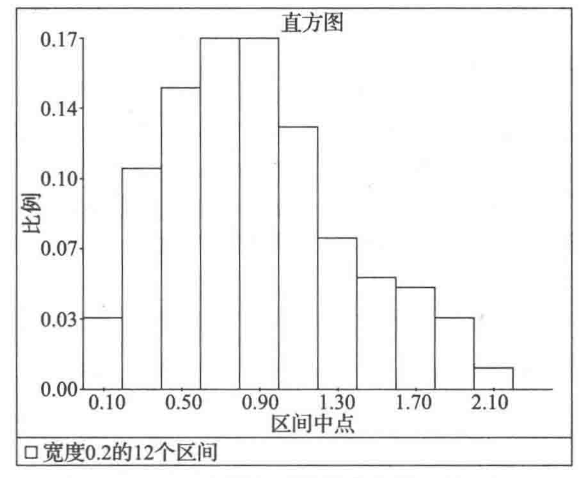


图 6.49　200 个服务时间的直方图，$\Delta b=0.2$

即使是一个分布排名第一，这也并不必然意味着该分布好到足以用于仿真中。然而，由于“绝对评价”为“好”，目前没有不使用韦布尔分布的证据。但是，谨慎的做法是使用“对比”模块获得更进一步的证实。如果排名最好的分布得到的绝对评价为“差”，那么它就不适合用于仿真模型，且 ExpertFit 会建议使用经验分布(参见第 6.2.4 小节)。

ExpertFit 排序评价算法如下。我们有 15 个启发式规则，它们具有某种能力来区分好的拟合分布和差的拟合分布(未考虑 χ^2 统计，因为它依赖于区间的选择)。为了确定这些启发式规则哪个实际上最好，从已知“父”分布中产生大小为 n 的随机样本，应用 15 个启发式规则的每一个，看看在实际上它是否能选出正确的分布。对 200 个独立样本重复做，对于给定的样本大小，给出每个启发式规则选择父分布的估计概率。对于 175 个父分布/样本大小对，重复整个过程，得到表现出众的若干启发式规则。这些启发式规则组合

起来形成整个算法以对拟合的分布进行排序，以及计算相对分数。生成的 35 000 个数据集也用于确定“绝对评价”的规则的开发。

备选模型的相关评价

模型	相对分数	参数	
1-韦布尔	100.00	位置	0.000 00
		比例	0.942 99
		形状	2.035 85
2-伽马	83.33	位置	0.000 00
		比例	0.271 97
		形状	3.076 19
3-韦布尔（E）	62.50	位置	0.065 52
		比例	0.851 73
		形状	1.702 30

分数在0.00与100.00之间的模型定义了7个

模型1-韦布尔的绝对评价
评价：好
建议：增加使用比较表评价会提供更多信息

关于模型1-韦布尔的附加信息
模型均值相对样本均值的“误差”

$-5.8399e-4 = 0.07\%$

图 6.50　服务时间数据的 ExpertFit 结果屏幕显示

排序屏幕也显示，韦布尔分布的均值与样本均值之间的误差只有 0.07%。

正如“绝对评价”中建议的一样，现在我们要通过图和拟合优良度检验来获得韦布尔分布进一步的信息。图 6.51 给出了韦布尔和伽马分布的密度直方图。可以看出，韦布尔分布很好地匹配了直方图，伽马分布明显要差一些。图 6.52 给出了两种分布的分布函数差图，再次可以看出韦布尔分布的优势。图 6.53 的 *P-P* 图也显示韦布尔分布更好些。

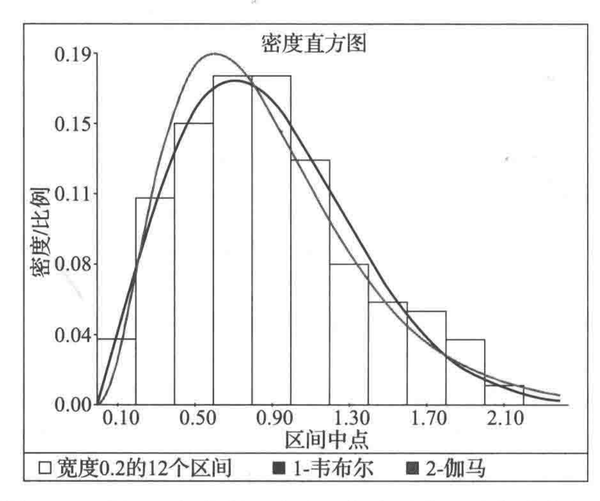


图 6.51　服务时间数据和韦布尔与伽马分布的密度直方图

我们下面对韦布尔分布做一个等概率 χ^2 检验，置信水平 $\alpha=0.05$，区间数为 $k=20$。

χ^2 统计量为 15.6，小于临界值 30.144；因此，这个特定的 χ^2 检验告诉我们没有理由拒绝韦布尔分布。K-S 检验的调整的统计量为 0.428，小于 $\alpha=0.05$ 的修正临界值 0.874。再次我们没有理由拒绝韦布尔分布。最后，A-D 检验统计量为 0.264，小于 $\alpha=0.05$ 的临界值 0.746，告诉我们也没有理由拒绝韦布尔分布。

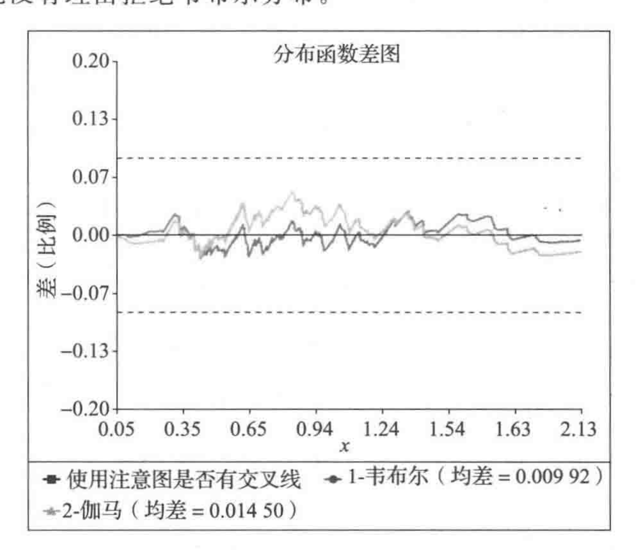


图 6.52 服务时间数据和韦布尔与伽马分布的分布函数差图

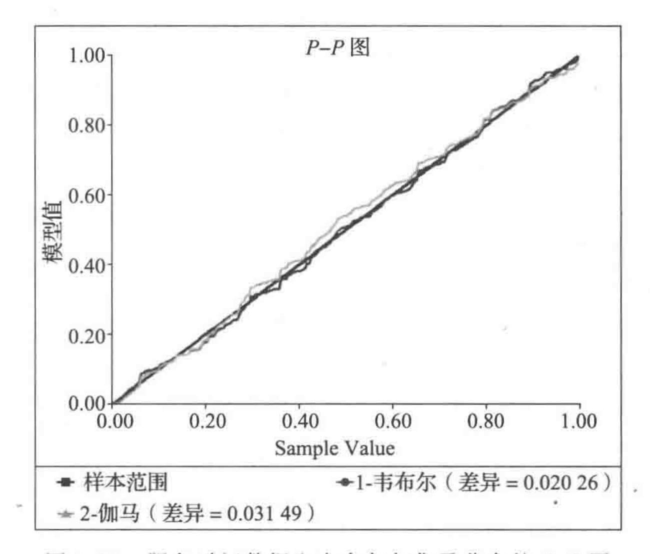


图 6.53 服务时间数据和韦布尔与伽马分布的 P-P 图

因此，基于“绝对评价”、图，以及拟合优良图检验，没有理由认为韦布尔分布不是服务时间数据的好的表示。

6.8 分布平移与截断

在第 6.2.2 小节讨论的指数分布、伽马分布、韦布尔分布、对数正态分布、皮尔逊Ⅴ型分布、皮尔逊Ⅵ型分布、对数逻辑斯蒂分布取值范围都是$[0, +\infty)$。因此，如果随机变量 X 具有以上这些分布的任何一种，它能取任意小的正数。然而实践中经常出现的是，如果 X 代表完成某任务所需的时间(例如顾客服务时间)，X 小于某个正数简直是不可能的。例如，在银行中服务任何人的时间恐怕都不可能，譬如说，小于 30s；这将反映在我

们收集的银行运营时间数据上。因此，在实际中 $P(X<30\text{s})=0$；然而，例如，对于一个拟合的伽马分布，存在产生小于 30s 的随机值正概率。因此，在某些情况下修改这些分布的形式会提供一个更加实际的模型并且会导致更好的拟合。

这个变化可以通过将分布向右平移某个距离实现。这实际等于是对所讨论的分布产生带位置参数的密度函数(参见第 6.2.1 小节)。例如，伽马分布向右平移任意量 $\gamma>0$ 后密度函数为：

$$f(x)=\begin{cases}\dfrac{\beta^{-\alpha}\,(x-\gamma)^{\alpha-1}\mathrm{e}^{-(x-\gamma)/\beta}}{\Gamma(\alpha)}, & x>\gamma\\ 0, & \text{其他}\end{cases}$$

它的形状和比例参数和 $\Gamma(\alpha,\ \beta)$ 分布一样，但是向右平移了 γ 单位(这也常常称为三参数伽马分布)。定义上面讨论的其他分布的平移版本是类似的，通过将密度函数中 x 更换为 $x-\gamma$，并更换相应的定义域。这些平移分布的范围为 $[\gamma,\ +\infty)$。

对于这些平移分布，我们然后需要估计 γ 以及其他参数。理论上讲，通过找到 γ 的 MLE 加上原有参数的 MLE，这就能做到。对于平移指数分布，$\hat{\gamma}$ 和 $\hat{\beta}$ 相对易于寻找(参见习题 6.12)。然而，寻找三参数分布的 MLE 明显是比较麻烦的。例如，在伽马分布、韦布尔分布和对数正态分布中，众所周知(全局)MLE 没有很好的定义[参见 Cheng 和 Amin(1983)，Cohen 和 Whitten(1980)，Zanakis(1979a)]。也就是说，选择 $\hat{\gamma}=X_{(1)}$(样本中的最小观测值)可能会使得似然函数 L 变为无穷大，这导致其他参数的无法取值。三参数估计问题的简单方法就是首先估计位置参数 γ，由

$$\widetilde{\gamma}=\frac{X_{(1)}X_{(n)}-X_{(k)}^2}{X_{(1)}+X_{(n)}-2X_{(k)}}$$

其中，k 为 $\{2,\ 3,\ \cdots,\ n-1\}$ 中的最小整数，使得 $X_{(k)}>X_{(1)}$[参见 Dubey(1967)]。

可以证明，当且仅当 $X_{(k)}<[X_{(1)}+X_{(n)}]/2$ 时 $\widetilde{\gamma}<X_{(1)}$；这是非常可能发生的，参见习题 6.23[Zanakis(1979b)实验给出了韦伯分布 $\widetilde{\gamma}$ 的准确度]。对于给定的 $\widetilde{\gamma}$ 值，我们定义 X_i' 为：

$$X_i'=X_i-\widetilde{\gamma}\geqslant 0,\quad 对于\ i=1,2,\cdots,n$$

最后，将通常的两参数 MLE 程序应用到观测值 X_1'，X_2'，…，X_n' 上，可以得到比例和形状参数的 MLE 估计。

例 6.21 图 6.54 给出了 $n=808$ 列煤车的卸载时间(小时)直方图($\Delta b=0.2$)，每列车大约有 110 个车厢(图 6.54 所示的实际上是密度直方图)。从直方图的形状可以看出，拟合的分布需要一个正的位置参数。由于 $X_{(1)}=3.37$，$X_{(2)}=3.68$，$X_{(808)}=6.32$，我们取 $\hat{\gamma}=3.329$。对于 $i=1,\ 2,\ \cdots,\ 808$，$X_i'=X_i-3.329$，将其用于计算 MLE，对于对数逻辑斯蒂分布可得 $\hat{\alpha}=7.451$，$\hat{\beta}=1.271$，其密度函数在图 6.54 中也给出了。整体上看，平移对数逻辑斯蒂分布和直方图吻合度是相当好的。◀

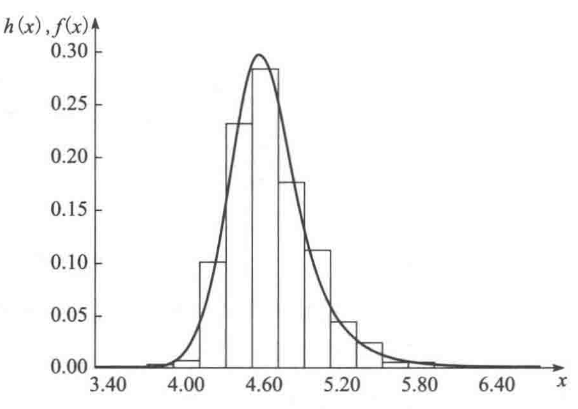


图 6.54 列车卸载时间数据密度直方图和拟合的对数逻辑斯蒂分布

在某些情况下，拟合的分布可能在整体上对观测数据提供了一个好模型，但是其他信息表明，例如，没有值能超过一个有限的常数 $b>0$。如果拟合的密度 f 的范围为 $[0,\ +\infty)$，这同上限 b 矛盾，因此我们会替代使用一个截断密度 $f^*(x)=f(x)/F(b)$ $(0\leqslant x\leqslant b)$

和 $f^*(x)=0$ (x 其他值)，其中，$F(b)=\int_0^b f(x)\mathrm{d}x<1$。从 f^* 中产生成随机值的方法在 8.2.1 小节中给出。

例 6.22 如果发现伽马分布对银行服务时间提供了一个好模型，密度函数可能在 $b>$ 15min 以上被截断，如果比该值更大的值极不可能出现的话。◀

6.9 贝塞尔分布

对于一组观测数据 X_1，X_2，…，X_n 建模确定概率分布有第 4 种方法(Wagner 和 Wilson(1996a，1996b)，其他的方法参见第 6.1 节)。如果 X 是在有限范围$[a, b]$上的一个连续随机变量，分布函数 $F(x)$具有任意形状，那么 $F(x)$可以被足够高阶 m 的贝塞尔(Bézier)分布函数任意逼近。令$\{\boldsymbol{p}_0, \boldsymbol{p}_1, \cdots, \boldsymbol{p}_m\}$为一组控制点，其中，$\boldsymbol{p}_i=(y_i, z_i)(i=1, 2, \cdots, m-1)$，$\boldsymbol{p}_0=(a, 0)$，$\boldsymbol{p}_m=(b, 1)$。贝塞尔分布函数 $\boldsymbol{P}(t)$通过下式参数化给出：

$$\boldsymbol{P}(t)=\sum_{i=0}^{m} B_{m,i}(t)\boldsymbol{p}_i, \quad t\in[0,1] \tag{6.6}$$

其中，

$$B_{m,i}(t)=\frac{m!}{i!(m-i)!}t^i(1-t)^{m-i}$$

令 $\boldsymbol{y}$ 为 y_i 的向量，并令 $\boldsymbol{z}$ 为 z_i 的向量。此外，令 $F_n(x)$为式(6.5)定义的经验分布函数，且令 $F(x; m, \boldsymbol{y}, \boldsymbol{z})$为式(6.6)给出的贝塞尔分布函数。对于固定的 m，通过使用合适的优化方法(例如最小二乘)，对于所有可能的 $\boldsymbol{y}$ 和 $\boldsymbol{z}$，在满足一定约束情况下，找到 $F_n(x)$ 和 $F(x; m, \boldsymbol{y}, \boldsymbol{z})$之间的最小距离，则 $F(x; m, \boldsymbol{y}, \boldsymbol{z})$是对 X_i 的拟合(优化确定$\boldsymbol{p}_i$，$i=1, 2, \cdots, m-1$)。

对于不能较好地用标准理论分布来表示的数据集合进行建模来说，贝塞尔分布是经验分布的替代方法。此外，Wagner 和 Wilson 开发了一个拟合贝塞尔分布的软件包。然而，在使用贝塞尔分布的时候有一个困难，至少在现在是，贝塞尔分布没有在大多数仿真包中实现，靠自己来做在某些软件中可能困难。

6.10 确定多元分布、相关性及随机过程

本章到目前为止，讨论的只是每次确定和估计单个、单随机变量的分布。如果仿真模型需要的输入只是标量随机变量，而且如果它们在模型之间相互独立，那么，对每个输入重复地应用本章中至今讨论的方法就够了。的确，这是在大多数仿真项目中的标准模式，因而它是大多数仿真软件包支持的模式。

然而，有些系统的输入随机变量之间以某种方式统计相关：

- 有些输入随机变量共同形成一个随机向量，具有某种多元(或联合)概率分布(参见第 4.2 节)，需要建模者确定。
- 在其他情况下，我们可能不想(也不能)走得如此之远，以确定整个多元分布，但虽然如此，在没有整个多元分布的知识或说明的情况下，疑问在于，具有自身或边际分布的不同的输入随机变量之间可能存在相关性。即使不能够确定整个多元分布，也会希望我们的输入反映出这种相关性。
- 还有一些情形，我们也许想要确定整个输入随机过程(参见第 4.3 节)，其中组成过程的单个随机变量的边际分布需要确定，以及它们之间的自相关经过某种要求的滞后之后表现出来。这可以看做无限维输入随机向量。

发生这种输入的实际物理环境是易于想到的：

- 考虑一个维修点，它可以建模成具有两个服务台的双队列系统。在第一个服务台，到来的零件被检查，对任何缺陷加以标记以便在第二个服务台维修。对检查和维修

来说，由于严重受损零件恐怕需要的时间都大于平均值，因此，我们会期望一个给定零件的两个服务时间是正相关的。Mitchell 等(1977)发现，在系统建模的时候忽视这个相关性会导致仿真结果的严重不准确。

- 在通信系统的模型中，到达报文的长度(可能还有报文之间的到达间隔时间)可形成一个随机过程，报文长度具有某种平稳的边际单变量分布，以及经过若干滞后表现出的某类相关性。例如，实际中可能是大报文倾向于成组到达，小报文也是这样，这导致报文长度输入过程内的正自相关性。Livny 等(1993)证明，简单的 $M/M/1$ 队列无论服务时间还是到达间隔时间输入过程的自相关对输出性能度量具有重要影响。
- 在库存或者生产系统模型中，如果一个周期的一个大订单倾向跟着下个周期的小订单，或者相反，到达订单流可能会显示出负的滞后一拍自相关性。

因此，如果建模人员有仿真的各种输入标量随机变量之间具有某种统计联系的证据，或者输入流在时间上呈现出自相关特征，那么仿真期间对这些关系建模并产生这些关系时就要给予注意，以避免可能的与模型确认有关的问题。

在本节剩余部分，将简要讨论有关确定和估计的上述某些问题，而在第 8.5 节，将讨论当仿真运行时为仿真输入产生相应观测值的问题。在第 6.12 节，将讨论对到达过程的建模相关问题；在第 8.6 节将讨论这种过程的产生方法。Leemis(2004)以及 Nelson、Yamnitsky(1998)的第 5 节中讨论了很多这方面问题。

6.10.1 确定多元分布

令 $\boldsymbol{X}=(X_1, X_2, \cdots, X_d)^{\mathrm{T}}$，维度为 d 的输入随机向量($\boldsymbol{A}^{\mathrm{T}}$ 表示向量或矩阵 $\boldsymbol{A}$ 的转置，$\boldsymbol{X}$ 为 $d\times1$ 的列向量)。例如，在上面的维修车间例子中，$d=2$(在这个例子中 $\boldsymbol{X}$ 称为双变量)，X_1 为零件的检查时间，X_2 为同一零件后续的维修时间。令 $\boldsymbol{X}_k=(X_{1k}, X_{2k}, \cdots, X_{dk})^{\mathrm{T}}$ 为 n 个独立同分布 d 维随机向量的第 k 个值；在维修车间例子中，对应于 n 个不同的零件检查和维修时间，有 n 对观测数据，即

$$\begin{pmatrix}X_{11}\\X_{21}\end{pmatrix},\begin{pmatrix}X_{21}\\X_{22}\end{pmatrix},\cdots,\begin{pmatrix}X_{1n}\\X_{2n}\end{pmatrix}$$

并想进一步能产生这种二维向量序列作为仿真的输入。

注意，当允许一个指定的 $\boldsymbol{X}_k$ 各个分量内部具有相关性时，这里假设不同 $\boldsymbol{X}_k$ 之间的分随机变量是相互独立的，即随机向量 $\boldsymbol{X}_1$，$\boldsymbol{X}_2$，$\cdots$，$\boldsymbol{X}_n$ 相互独立。在维修车间的情形，这意味着某个零件的检查和维修时间可以是相关的，但是在不同零件检查时间之间，以及维修时间之间没有关系——假设零件彼此是独立的[在第 6.10.3 小节中，允许序列$\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots\}$中自相关，但是大多数都是在单变量(标量)$d=1$ 的情况下]。

对于任意固定的 d 维向量 $\boldsymbol{x}=(x_1, x_2, \cdots, x_d)^{\mathrm{T}}$，随机向量 $\boldsymbol{X}$ 的多元(联合)分布函数定义为：

$$F(\boldsymbol{x})=P(\boldsymbol{X}\leqslant\boldsymbol{x})=P(X_1\leqslant x_1, X_2\leqslant x_2, \cdots, X_d\leqslant x_d)$$

这包含了所有连续、离散、混合的单一边际分布的情况。除了隐含有边际分布之外，关于单一随机变量分量之间的关系的所有信息都包含在多元分布函数中，包括相关性。

正如处理单一标量随机变量那样，有各种多元分布已经开发出来，并进行了参数化，有各种方式并进行了各种分类；参见 Devroye(1986)的第Ⅺ章，约翰逊(1987)、约翰逊等(1997)、Kotz 等(2000)。然而，从观测数据来估计随机向量的整个多元分布是困难的，或者是不实际的，特别是如果样本量 n 不大的话更是如此。因此，在本小节剩余部分限制在仿真中已经发现是有用的多元分布估计的某些特定情形，并且在第 6.10.2 小节讨论如果估计目标比确定整个多元分布更适当时候能做什么。而且由于我们的兴趣最终在仿真输入上，也必须关注如何实现这种随机向量的产生，正如在第 8.5 节中讨论的那样。

多元正态分布

这可能是多元分布中最特殊的情形。虽然由于它的边际分布对称，并且双向具有无限

的尾部，这种分布在直接应用于仿真输入模型时有一些限制，然而它用作其他更有用的输入建模的分布的跳板。

对于 d 维实数空间的任意向量 $\boldsymbol{x}$，多元正态密度函数(参见第 4.2 节)定义为：

$$f(\boldsymbol{x}) = (2\pi)^{-n/2} \ |\boldsymbol{\Sigma}|^{-1/2} \exp\left[-\frac{(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}{2}\right]$$

这里，$\boldsymbol{\mu}=(\mu_1, \mu_2, \cdots, \mu_d)^{\mathrm{T}}$ 为均值向量；$\boldsymbol{\Sigma}$ 为协方差矩阵，第(i, j)项为 $\sigma_{ij}=\sigma_{ji}=\mathrm{cov}(X_i, X_j)$(因此 $\boldsymbol{\Sigma}$ 是对称正定的)，$|\boldsymbol{\Sigma}|$ 为 $\boldsymbol{\Sigma}$ 的行列式，$\boldsymbol{\Sigma}^{-1}$ 为 $\boldsymbol{\Sigma}$ 的逆矩阵。X_i 的边际分布为 $N(\mu_i, \sigma_{ii})$。我们记多元正态分布为 $N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$。

X_i 和 X_j 之间的相关系数为 $\rho_{ij}=\rho_{ij}/\sqrt{\rho_{ii}\rho_{jj}}=\rho_{ji}$，始终在 $-1\sim+1$ 之间。因此，由于 $\sigma_{ij}=\sigma_{ji}=\rho_{ij}\sqrt{\sigma_{ii}\sigma_{jj}}$，另一个参数化多元正态分布就是用参数 σ_{ij} 和 ρ_{ij} 替换 $\boldsymbol{\Sigma}$，其中 $i=1, 2, \cdots, d$，$j=i+1, i+2, \cdots, d$。

注意，在多元正态情形，整个联合分布唯一地由边际分布和相关系数来确定。这个性质对于其他的非多元正态分布一般不成立；即可能有多种不同的联合分布得到同样一组边际分布和相关系数。

为了对 d 维观测数据 $\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n$ 拟合多元正态分布，通过 MLE 来估计均值向量 $\boldsymbol{\mu}$：

$$\hat{\boldsymbol{\mu}} = \overline{\boldsymbol{X}} = (\overline{X}_1, \overline{X}_2, \cdots, \overline{X}_d)^{\mathrm{T}} \tag{6.7}$$

其中，$\overline{X}_i = \sum_{k=1}^{n} X_{ik}/n$。

协方差矩阵 $\boldsymbol{\Sigma}$ 通过 $d\times d$ 的矩阵$\boldsymbol{\Sigma}$来估计，它的第(i, j)项为

$$\hat{\sigma}_{ij} = \frac{\sum_{k=1}^{n}(X_{ik}-\overline{X}_i)(X_{jk}-\overline{X}_j)}{n} \tag{6.8}$$

相关系数 ρ_{ij} 通过 MLE 估计得到：

$$\hat{\rho}_{ij} = \frac{\hat{\sigma}_{ij}}{\sqrt{\hat{\sigma}_{ii}\ \hat{\sigma}_{jj}}} = \hat{\rho}_{ji} \tag{6.9}$$

对于通过这种方法估计的多元正态分布，从它产生的输入随机变量可采用在第 8.5.2 小节给出的一些方法得到。

多元对数正态分布

这个多元分布能够为建模人员提供在$[0, +\infty)$上正偏的相互之间可能相关的边际分布。

较之给出其完整联合密度函数的严格定义而言，讨论多元对数正态分布到多元正态分布之间的转换关系对仿真来说更为有用。称 $\boldsymbol{X}=(X_1, X_2, \cdots, X_d)^{\mathrm{T}}$ 具有多元对数正态分布，当且仅当：

$$\boldsymbol{Y} = (Y_1, Y_2, \cdots, Y_d)^{\mathrm{T}} = (\ln X_1, \ln X_2, \cdots, \ln X_d)^{\mathrm{T}}$$

具有多元正态分布 $N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$；参见 Jones 和 Miller(1966)以及 Johnson 和 Ramberg (1978)。换句话说，多元对数随机向量 $\boldsymbol{X}$ 可以表示成：

$$\boldsymbol{X} = (\mathrm{e}^{Y_1}, \mathrm{e}^{Y_2}, \cdots, \mathrm{e}^{Y_d})^{\mathrm{T}}$$

其中，$\boldsymbol{Y}$ 为多元正态分布 $N_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$。

X_i 的边际分布为一元对数正态分布 $LN_d(\mu_i, \sigma_{ii})$，其中，μ_i 为 $\boldsymbol{\mu}$ 的第 i 个元素；σ_{ii} 为 $\boldsymbol{\Sigma}$ 中的第 i 个对角项。

由于以 $\boldsymbol{X}$ 的上述对数变换的多元正态随机向量 $\boldsymbol{X}$ 具有均值向量 $\boldsymbol{\mu}=(\mu_1, \mu_2, \cdots, \mu_d)^{\mathrm{T}}$ 以及协方差矩阵 $\boldsymbol{\Sigma}$，其第(i, j)项为 σ_{ij}(因此，相关系数为 $\rho_{ij}=\sigma_{ij}/\sqrt{\sigma_{ii}\sigma_{jj}}$)。因此得到：

$$E(X_i) = \mathrm{e}^{\mu_i+\sigma_{ii}/2} \tag{6.10}$$

$$\mathrm{var}(X_i) = \mathrm{e}^{2\mu_i+\sigma_{ii}}(\mathrm{e}^{\sigma_{ii}}-1) \tag{6.11}$$

并且

$$\mathrm{cov}(X_i, X_j) = (\mathrm{e}^{\sigma_{ij}} - 1)\exp\left(\mu_i + \mu_j + \frac{\sigma_{ii} + \sigma_{jj}}{2}\right) \tag{6.12}$$

这意味着 X_i 和 X_j 之间的相关系数为：

$$\mathrm{cor}(X_i, X_j) = \frac{\mathrm{e}^{\sigma_{ij}} - 1}{\sqrt{(\mathrm{e}^{\sigma_{ii}} - 1)(\mathrm{e}^{\sigma_{jj}} - 1)}} \tag{6.13}$$

注意，这里的 $\boldsymbol{\mu}$ 以及 $\boldsymbol{\Sigma}$ 不是多元对数正态随机向量 $\boldsymbol{X}$ 的均值向量和协方差矩阵，而是对应多元正态随机向量 $\boldsymbol{Y}$ 的均值和方差。$\boldsymbol{X}$ 的均值向量和协方差矩阵(以及相关系数)由上述式(6.10)～式(6.13)给出。

为了对 d 维空间向量的样本 $\boldsymbol{X}_1$，$\boldsymbol{X}_2$，…，$\boldsymbol{X}_n$ 拟合多元对数正态分布，对每个观测数据向量的每个分标量观测值取自然对数以得到数据向量 $\boldsymbol{Y}_1$，$\boldsymbol{Y}_2$，…，$\boldsymbol{Y}_n$；将这些 $\boldsymbol{Y}_k$ 值作为具有未知均值向量 $\boldsymbol{\mu}$ 和协方差矩阵 $\boldsymbol{\Sigma}$ 的多元正态分布对待，通过上述式(6.7)和式(6.8)分别估计 $\boldsymbol{\mu}$ 和 $\boldsymbol{\Sigma}$。

从拟合的多元对数正态分布中生成输入随机变量方法在第 8.5.2 小节讨论。

多元约翰逊变换系统

单变量约翰逊变换系统包括了正态分布、对数正态分布、约翰逊 S_B(参见第 6.2.2 小节)、约翰逊 S_U 分布，在拟合分布的范围和形状上提供了很可观的灵活性。该分布族已经扩展到多元情形；参见 Johnson(1987)的第 5 章、Stanfield 等(1996)以及 Wilson(1997)。

正如在一元情形一样，多元约翰逊变换系统在拟合的范围和形状上允许很大的灵活性，对广泛范围变化的所观测的多元数据向量获得很好的拟合，它明显地比上面讨论的多元正态分布或者对数正态分布要灵活得多。特别是，Stanfield 等(1996)研究了一种方法，使得拟合多元分布的边际分布的前 4 阶矩和观测数据相匹配，拟合分布的相关性结构和经验分布匹配得一样好。对观测数据拟合这种分布包括若干步骤，且使用拟合一元约翰逊分布的方法；有关这种拟合步骤的细节，以及如何从拟合多元约翰逊分布中产生随机向量，参见 Stanfield 等(1996)。

二元贝塞尔分布

正如在第 6.9 节讨论的，Wagner 和 Wilson(1995)将的一元贝塞尔分布扩展到二元情况(d=2 维)，其中还介绍了允许进行图形化交互调整拟合分布的软件；此外，还讨论了生成随机向量的方法。有关二元贝塞尔分布更多的结果和方法可以在 Wagner 和 Wilson(1996a)找到。

Wagner 和 Wilson(1995)说明将贝塞尔分布扩展到三维以及三维以上是“可行但是很烦琐”。

6.10.2 确定任意边际分布与相关性

在第 6.10.1 小节讨论的几种情形是确定一个完整多元分布来对 d 个可能的相关输入随机变量之间的联合行为进行建模，这些随机变量一起构成一个输入随机向量。在这些每一种情形，涉及的多元分布族的拟合成员(正态分布、对数正态分布、约翰逊分布和贝塞尔分布)确定了向量内分随机变量对之间的相关性以及其边际分布；拟合成员还将分随机变量之间的联合方差作为联合密度函数本身一部分进行了较通用而完整的描述。

有时候我们需要比这种做法更大的灵活性。我们也许想要允许我们的仿真模型的输入随机变量各对之间具有相关性，而又不强行对同族成员的全体拟合边际分布以得到整体多元随机分布。换言之，我们想要自由地确定任意一元分布以分别对每个输入随机变量建模，如在第 6.1 节～第 6.9 节中所描述的那样，还要不用它们的边际分布来估计它们之间的相关性。实际上，希望某些分输入随机变量是连续的，而其他为离散的，以及甚至其他为连续-离散混合的，还有甚至允许这些变量之间具有相关性。

做这件事情的一个非常简单而且相当直观的方法就是对所包括的每个一元随机变量拟

合随机变量，一次一个并同其他变量隔离，然后用式(6.9)来估计输入随机变量对之间可能的相关性。将这些随机变量组成一个随机向量，那么这个向量在结构上就具有了所要求的一元边际分布和所要求的相关性结构。然而，重要的是要注意，这种方法并不确定或者“控制”得出整个随机向量的联合分布——实际上，我们甚至一般都不会知道这种联合分布是什么。因此，该建议的方法对边际分布和相关性允许更大的灵活性，但是同时也消弱了整体控制。另外一个需要注意的地方就是边际分布的形式和参数会限制某些可能的相关性，参见 Whitt(1976)。

虽然这种情况的确定方法看起来相对直观，至少在原理上如此，然而也必须确保这里确定的任何分布在仿真时可由它产生。这点将在第 8.5.5 小节中讨论，它基于 Hill 和 Reilly(1994)和 Cario 等(2002)的工作。

6.10.3 确定随机过程

正如早前提及过的，有这样一种情况，同一物理对象的输入随机变量序列适于建模为来自相同的(边际)分布，而在序列内部随机变量之间可能呈现出某种自相关性。例如，如果$\{X_1, X_2, \cdots\}$表示到达通信节点的连续报文的大小，X_i 可能来自相同的(平稳)分布，但是对于滞后 $l=1, 2, \cdots, p$，$\text{cov}(X_i, X_{i+1})$可能非零，其中，作为建模活动的一部分，要确定最大的滞后 p 的自相关系数。在这种情况下中，X_i 是同分布的，但是它们并不独立，因此构成了一个平稳随机过程，可能具有滞后为 p 的自相关。正如早前提及过的，这种在输入流内的自相关能对仿真结果产生重要的影响，正如 Livny 等(1993)论证的那样。

本小节简要地介绍这种情况的某些模型，而在第 8.5.6 小节将讨论如何能从这种模型产生，实现仿真的输入。除了 VARTA 过程外，只考虑过程中点 X_i 为一元(标量)随机变量的情况，而不是本身就是多元随机向量的情况。

AR 和 ARMA 过程

为进行时间序列数据分析，Box 等(1994)研究出了标准自回归(AR)或者自回归滑动平均(ARMA)模型，该模型最初考虑对输入时间序列建模。虽然文献中关于这些过程有很多不同的参数化方法，均值为 μ 的平稳 $AR(p)$的一个版本是：

$$X_i = \mu + \phi_1(X_{i-1} - \mu) + \phi_2(X_{i-2} - \mu) + \cdots + \phi_p(X_{i-p} - \mu) + \varepsilon_i \tag{6.14}$$

其中，ε_i 为均值为 0 的 IID 正态随机变量，ε_i 的方差选择用于控制 $\text{var}(X_i)$；在满足 X_i 具有平稳边际分布的条件下，ϕ_i 为常数。ARMA 模型的定义对上面的递归表达式中 ε_i 的过去的值增加了权系数[完整内容参见 Box 等(2008)]。

为了对观测数据拟合这种模型，采用线性回归方法来估计过程中的未知参数。X_i 的边际分布一般限制为正态分布，然而这也限制了模型在仿真输入建模中直接使用，因为范围在双向都是无限的。然而 AR 过程确实奠定了下面将要讨论的 ARTA 模型过程的基础，作为仿真输入过程模型，ARTA 模型更为灵活且更为有用。

伽马过程

Lewis 等(1989)研究了这些过程，产生出具有伽马分布的边际分布，以及过程内部点之间的自相关系数。这些过程是通过一种自回归操作来构建的，同上面式(6.14)中所描述的正态 AR 过程的思想类似。它包括了指数边际分布的情况，也叫做指数自回归(EAR)过程。

ARTA 过程

Cario 和 Nelson(1996)提出了任意自回归(ARTA)过程，与 TES 过程一样，它试图对任意平稳边际分布以及任意自相关结构建模。ARTA 过程能够精确地将所要求的自相关结构产生出指定的滞后 p，以及所要求的边际分布函数；此外，它们都是通过自动程序来确定的，不需要主观的交互式操作。

为了定义一个 ARTA 过程，先确定一个标准平稳 AR 过程$\{Z_i\}$，其边际分布为 $N(0, 1)$($\{Z_i\}$称为基过程)。然后定义到仿真的最终输入过程为：

$$X_i = F^{-1}[\Phi(Z_i)] \tag{6.15}$$

其中，F^{-1}为所要求的平稳边际分布函数 F 的逆；Φ 指 $N(0, 1)$分布函数。因为根据熟知的概率积分变换的基本结果[参见 Mood、Graybill 和 Boes，(1974，第 202-203 页)]$\Phi(Z_i)$具有 $U(0, 1)$分布，对该 $U(0, 1)$随机变量应用 F^{-1}运算就得到具有分布函数 F 的随机过程。因此，显然，X_i 的边际分布就是所要求的 F。

然而，确定所要求的 ARTA 过程的原理性工作是确定基过程$\{Z_i\}$的自相关结构，使得最终输入过程$\{X_i\}$能够呈现出所要求的自相关结构。Cario 和 Nelson(1998)开发了这样做的数值方法以及执行计算的软件包。该软件假设$\{X_i\}$过程的边际分布和自相关系数是已知的，哪怕是如果要求的话，它将由一组观测的时间-序列数据来计算出样本自相关系数。

Biller 和 Nelson(2005，2008)提出了一种统计方法，用于对一组观测的一元时间-序列数据从约翰逊变换系统拟合一种具有边际分布的 ARTA 过程(参见第 6.10.1 小节)。

VARTA 过程

Biller 和 Nelson(2003)提出了对平稳多元随机过程$\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots\}$建模并产生它的一种方法，称为向量任意自回归(VARTA)过程。对于 $i=1, 2, \cdots$，令 $\boldsymbol{X}_i=\{X_{1i}, X_{2i}, \cdots, X_{di}\}^{\mathrm{T}}$ 为时间 i 时维数为 d 的输入随机向量。对于 $i=1, 2, \cdots, j=1, 2, \cdots, d$，令 F_j 为 $X_{j,i}$的分布函数。而且，对于 j，$k=1, 2, \cdots, d$，以及滞后 $l=0, 1, \cdots, p$，令 $\rho_{j,k,l}(\boldsymbol{X})=\mathrm{cor}(X_{j,i}, X_{k,i+l})$，其中，$\rho_{j,j,0}(\boldsymbol{X})=1$。这种方法假设 F_j 为约翰逊变换系统的一个给定的成员，并且相关系数 $\rho_{j,k,l}(\boldsymbol{X})$也已经确定(一般情况下，对于每个 j 值，F_j 将是不一样的)。

确定所要求的 VARTA 过程的原理性工作就是确定高斯向量自回归基过程$\{\boldsymbol{Z}_i\}$的自相关结构，使得得到的最终输入过程$\{\boldsymbol{X}_i\}$呈现出所要求的自相关结构。

Biuer(2009)使用 copula 理论推广了 VARTA 过程，以表示这种情况下出现的依赖结构，其中极端分量实现一起出现。

6.11 缺少数据时分布的选择

在一些仿真研究中，不可能对所关心的随机变量收集到数据，使得第 6.4 节到第 6.6 节的方法不适用于选择对应概率分布的问题。例如，如果研究的系统目前尚未以某种形式存在，那么显然不可能从系统中搜集数据。这个困难对现存系统也会出现，倘若所需要的概率分布数目特别多，仿真研究有时间限制而不能够进行必需的数据收集和分析。此外，有时候数据是通过自动数据收集系统收集的，这些系统并不能以合适的格式提供数据。本节讨论数据缺失的情况下选择分布的四种启发式方法。

假设所关心的随机变量为连续随机变量 X。考虑这个随机变量是完成某项任务的时间也将是有用的，例如，当设备故障停机时修理设备的零件所需要的时间。使用三角分布或者贝塔分布方法的第一个步骤就是辨识出区间$[a, b]$(其中 a 和 b 为实数，并且 $a<b$)，使人感觉到 X 以接近概率 1 落在该区间内；也就是，$P(a\leqslant x\leqslant b)\approx 1$。为了得到 a 和 b 的主观估计，询问领域专家(SME)有关完成任务时间的他们最乐观和最悲观的估计分别是什么。一旦区间$[a, b]$被主观地辨识出来，下一步就是将概率密度函数放在$[a, b]$上，它被认为是 X 的代表。

在三角分布方法中，还要询问对于 SME 有关完成任务最可能的时间他们的主观估计。这个最可能值 m 就是 X 的分布的众数。给定 a、b 和 m，随机变量 X 就看做为具有$[a, b]$区间上的三角分布(参见第 6.2.2 小节)，众数为 m。图 6.17 给出了三角分布密度函数的图形。此外，产生三角随机变量的算法将在第 8.3.15 小节中给出。

三角方法的一个困难之处就是它需要主观估计绝对最小值和最大值 a、b，这是个问题。例如，b 值是未来 3 个月的最大值还是整个寿命期的最大值？三角分布的第二个主要问题是它不能有长右尾，而完成某项任务的时间的密度函数却往往有长右尾这种情况[Keefer 和 Bodily(1983)讨论了三角分布的其他形式]。

将密度函数放在[a，b]上的第二种方法就是假设随机变量 X 在该区间上具有贝塔分布(参见第 6.2.2 小节)，具有形状参数 α_1，α_2。由于贝塔分布密度函数能够假设的形状各种各样(参见图 6.11)，因此这种方法提供了更大的建模灵活性。另一方面，为了完全确定该分布还必须挑选 α_1，α_2，如何选择还不清楚。如果希望假设 X 在 a、b 之间取任何值的概率都是相同的，那么令 $\alpha_1=\alpha_2=1$，其结果就是 $U(a, b)$分布(参见图 6.11，如果对随机变量 X 除了范围[a，b]之外知之甚少，可以用这个模型)。另外一个主意，我们觉得通常更加实际，就是假设 X 密度函数向右偏移(我们以实际数据的经验表明，与任务完成时间相应的密度函数往往有这个形状)。这种密度形状在贝塔分布中对应于 $\alpha_1>\alpha_2>1$(参考图 6.11)。此外，这样的贝塔分布函数具有的均值 μ，众数 m 分别为：

$$\mu=a+\frac{\alpha_1(b-a)}{\alpha_1+\alpha_2},\quad m=a+\frac{(\alpha_1-1)(b-a)}{\alpha_1+\alpha_2-1}$$

给定 μ 和 m 的主观估计，可求解这些方程以得到下述的 α_1 和 α_2 的估计：

$$\widetilde{\alpha}_1=\frac{(\mu-a)(2m-a-b)}{(m-\mu)(b-a)},\quad \widetilde{\alpha}_2=\frac{(b-\mu)\,\widetilde{\alpha}_1}{\mu-a}$$

然而，要注意，对于密度向右偏的，μ 必须大于 m；如果 $\mu<m$，那么它将会向左偏。产生贝塔分布随机变量的算法在第 8.3.8 小节中给出。

确定贝塔分布的第二困难是，有些 SME 难于区分分布的均值和众数。Keefer 和 Bodily(1983)建议使用其他的方法来确定贝塔分布的参数。

人们有时使用三角分布或者贝塔分布来对随机性进行建模，即使可以收集并分析必要数据。这样做恰恰是因为分析人员不想麻烦地收集数据，或者分析人员不明白挑选合适分布的重要性。例 6.23 说明了不恰当使用三角(贝塔)分布有时会导致十分错误的结果。

例 6.23 考虑一个单服务台队列系统，具有指数到达间隔时间，均值为 1，与对数正态服务时间，均值为 0.9，方差为 1.39($\mu=-0.605$ 且 $\sigma^2=1$)，如图 6.55所示。但是，对分析人员来说服务时间分布实际是不知道的，他首先试图采用$a=0$、$m=0.2$、$b=1.97$ 的三角分布来近似服务时间分布。注意，这里 a 和 m 都猜对了。使用附录 1B 中 $M/G/1$ 队列中稳态平均延误 d 的公式，可以看出对于对数正态分布，$d=11.2$，而对于该三角分布，$d=1.3$。因此，用该三角分布来近似对数正态分布将会带来 88.2%的误差(参见表 6.20 和图 6.55)。

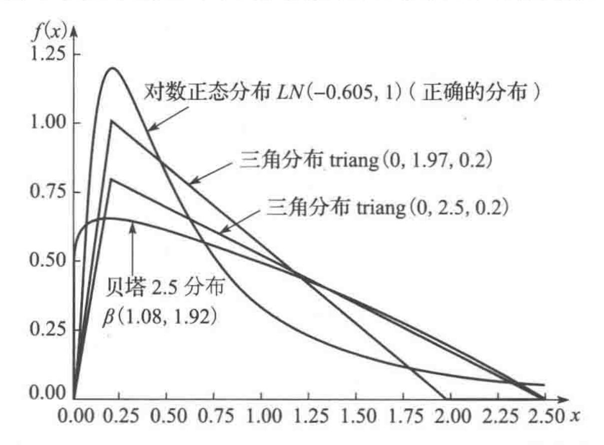


图 6.55　对数正态分布以及近似的三角分布和贝塔分布

表 6.20　用三角分布或者贝塔分布近似对数正态分布

服务时间分布	队列中稳态平均延误时间 d	错误百分比
对数正态分布 $LN(-0.605, 1)$	11.02	0
三角分布 triang(0, 1.97, 0.2)	1.30	88.2
三角分布 triang(0, 2.5, 0.2)	5.60	48.7
贝塔 2.5 分布 $\beta(1.08, 1.92)$	5.85	46.9

或者，假设分析人员试图使用 $a=0$、$m=0.2$，而均值为 0.9(正确)的三角分布来近似该未知的对数正态分布，结果为 $b=2.5$[三角分布的均值为(a+b+m)/3]。在这种情形下，$d=5.66$，仍然有 48.7%的误差。

最后，假设分析人员试图使用参数 $a=0$，$b=2.5$，$\mu=0.9$，$m=0.2$(和第二个三角分布相同)的贝塔分布来近似对数正态分布，结果为 $\widetilde{\alpha}_1=1.08$，$\widetilde{\alpha}_2=1.92$。在这种情形下，$d=5.85$，有 46.7%的误差。

总之，我们已经看到，使用三角分布或者贝塔分布来近似一个未知分布可能导致很大误差的仿真输出。 ◀

* 由于三角分布和贝塔分布的这些缺点，我们现在提出在数据缺失的情况下表示任务时间的两个新的模型，它们都基于对数正态分布和韦布尔分布。这些模型需要对任务时间分布的位置参数 γ、最大可能任务时间 m，以及 q 分位点($100q$ 百分点)进行主观估计。位置参数 γ 所起的作用类似于上面的最小值 a，但是现在 X 必须大于 γ。我们还假设 $0\leqslant\gamma<m<x_q<+\infty$。

我们从对数正态分布开始。如果 Y 是均值为 μ，方差为 σ^2 的正态分布，那么 $V=e^Y$ 是(双参数)对数正态分布，比例参数为 e^μ($e^\mu>0$)(参见 6.2.2 小节)，形状参数为 σ($\sigma>0$)。如果 $X=V+\gamma$，那么 X 是三参数对数正态分布(参见第 6.8 节)，位置参数为 γ，比例参数为 e^μ，形状参数为 σ，记作 $LN(\gamma, \mu, \sigma^2)$。从第 6.2.2 小节中关于对数正态分布的讨论可以得出 X 的众数为：

$$m=\gamma+e^{\mu-\sigma^2} \tag{6.16}$$

进一步，易于看出(参见习题 6.28)：

$$x_q=\gamma+e^{\mu+z_q\sigma} \tag{6.17}$$

其中，z_q 为 $N(0, 1)$随机变量的 q 分位点。

如果将式(6.16)中的 e^μ 代入到式(6.17)中，那么得到下面 σ 的二次方程：

$$\sigma^2+z_q\sigma+c=0$$

其中，$c=\ln[(m-\gamma)/(x_q-\gamma)]<0$。对 σ 求解该方程，可以得到下面的 σ 表达式：

$$\sigma=\frac{-z_q\pm\sqrt{z_q^2-4c}}{2}$$

由于 σ 必须为正数，我们取“+”的根，那么形状参数 σ 的估计 $\widetilde{\sigma}$ 就是：

$$\widetilde{\sigma}=\frac{-z_q+\sqrt{z_q^2-4c}}{2} \tag{6.18}$$

将 $\widetilde{\sigma}$ 代入到式(6.16)中，得到 μ 的估计 $\widetilde{\mu}$ 为：

$$\widetilde{\mu}=\ln(m-\gamma)+(\widetilde{\sigma})^2 \tag{6.19}$$

例 6.24 假设想要一个位置参数 $\gamma=1$、最大可能值 $m=4$，0.9 分位点(第 90 个百分点)$x_{0.9}=10$。从式(6.18)到式(6.19)可以得到 $\widetilde{\sigma}=0.588$，$\widetilde{\mu}=1.444$，图 6.56 给出了对数正态密度函数。 ◀

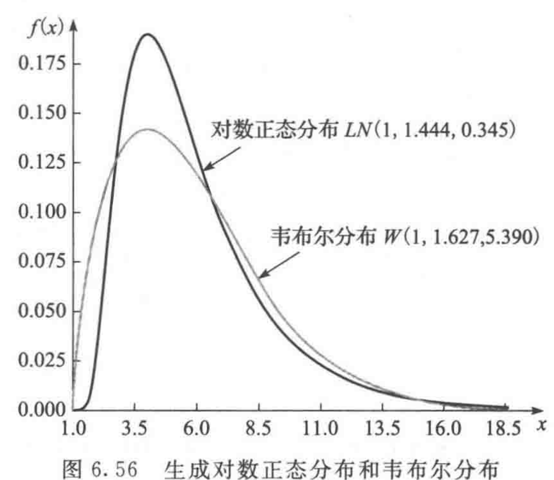


图 6.56　生成对数正态分布和韦布尔分布

现在考虑韦布尔分布。假设随机变量 Y 是一个(双变量)韦布尔分布，形状参数为 α($\alpha>0$)和比例参数为 β($\beta>0$)。进一步假设 $\alpha>1$，使得众数大于 0。如果 $X=Y+\gamma$，那么 X 为三参数韦布尔分布(参见第 6.8 节)，位置参数为 γ，形状参数为 α，比例参数为 β，记作 $W(\gamma, \alpha, \beta)$。X 的众数为(参见第 6.2.2 小节)：

---

* 第一次阅读时可以跳过本节剩余的部分。

$$m=\gamma+\beta\left(\frac{\alpha-1}{\alpha}\right)^{1/\alpha}$$

可重写该式为：

$$\beta=\frac{m-\gamma}{[(\alpha-1)/\alpha]^{1/\alpha}} \tag{6.20}$$

进一步，X 的分布函数在 x_q 处的取值为 $F_x(x_q)$，由下式给出(参见第 6.8 节)：

$$F_x(x_q)=1-e^{-[(x_q-\gamma)/\beta]^{\alpha}}=q$$

重写该式为：

$$\beta=\frac{x_q-\gamma}{\{\ln[1/(1-q)]\}^{1/\alpha}} \tag{6.21}$$

等式(6.20)和式(6.21)给出 α 的下述表达式：

$$\frac{m-r}{x_q-\gamma}=\left\{\frac{\alpha-1}{\alpha\ln[1/(1-q)]}\right\}^{1/\alpha} \tag{6.22}$$

该方程不能闭式求解，但是可以使用牛顿法迭代求解[参见 Press 等(1992)]以得到形状参数 α 的估计值 $\widetilde{\alpha}$(参见习题 6.29)。那么可将 $\widetilde{\alpha}$ 代入到式(6.20)中就可以得到比例参数 β 的估计 $\widetilde{\beta}$：

$$\widetilde{\beta}=\frac{m-\gamma}{[(\widetilde{\alpha}-1)/\widetilde{\alpha}]^{1/\alpha}} \tag{6.23}$$

例 6.25 假设想要一个韦布尔分布，位置参数 $\gamma=1$，最大可能值 $m=4$，0.9 分位值为 $x_{0.9}=10$。由式(6.22)和式(6.23)，我们得到，$\widetilde{\alpha}=1.627$，$\widetilde{\beta}=5.390$，得到结果韦布尔分布密度函数也在图 6.56 中给出。注意估计值 $\widetilde{\alpha}$ 和 $\widetilde{\beta}$ 是使用ExpertFit计算出来的(参见第 6.7 节)。◀

对数正态分布和韦布尔分布可以取任意大的值，虽然概率非常小。因此，如果知道对应随机变量不可能取值大于 $b(b>x_q)$，那么也许要求对该分布函数在 b 处截断(参见第 6.8 节)。

注意，基于 a(最小值)、m 和 x_q 的主观估计，也能确定一个三角分布(参见习题 6.30)。

6.12　到达过程模型

在很多仿真中，需要按时间 $0=t_0\leqslant t_1\leqslant t_2\leqslant\cdots$，产生随机点的序列，使得第 i 个某类事件在时间点 $t_i(i=1, 2, \cdots)$ 发生，而且事件时间 $\{t_i\}$ 的分布服从某个规定的形式。令 $N(t)=\max\{i: t_i\leqslant t\}$ 为 t 时刻或以前发生的事件的个数。称随机过程 $\{N(t), t\geqslant 0\}$ 为到达过程，是因为，感兴趣的事件通常是顾客到达某种服务设施。顺着这种说法，称 $A_i=t_i-t_{i-1}$(其中 $i=1, 2, \cdots$)是第 $i-1$ 个和第 i 个顾客之间的到达间隔时间。

在第 6.12.1 小节讨论泊松过程，该到达过程的 A_i 为独立同分布(IID)指数随机变量。泊松过程恐怕是对顾客到排队系统的到达时间最常用的模型。第 6.12.2 小节讨论非平稳泊松过程，常常用做到达系统的速率随时间变化的到达过程模型。最后，在第 6.12.3 小节描述每个事件实际上是顾客按“批”到达的到达过程进行建模的方法。

关于本节一般的参考文献是 Cinlar(1975，第 4 章)。

6.12.1　泊松过程

在本节定义一个泊松过程，说明它的一些重要性质，这样做的目的是解释很多实际系统的到达间隔时间十分接近 IID 指数随机变量的原因。

称随机过程 $\{N(t), t\geqslant 0\}$ 为泊松过程，如果：

(1) 顾客每次到达一个。

(2) $N(t+s)-N(t)$(在时间区间 $(t, t+s]$ 上到达的个数)独立于 $\{N(u), 0\leqslant u\leqslant t\}$。

(3) 对于所有 t，$s\geqslant 0$，分布 $N(t+s)-N(t)$ 独立于 t。

性质(1)和(2)是很多实际到达过程的特征。如果顾客成批到达，性质(1)不成立；参

见第 6.12.3 小节。性质(2)表示在区间$(t, t+s]$上的到达数与之前的时间区间$[0, t]$上的到达数目无关，并且与这些到达发生的时间也无关。若在$[0, t]$有很多到达事件，导致在$(t, t+s]$到达的某些顾客被阻塞，即未曾服务就立即离开，因为他们发现系统高度拥挤，像这样的例子，就会违反性质(2)。另一方面，很多实际生活的到达过程会违反性质(3)，因为它意味着顾客到达的速率不依赖于一天中的时间，等等。但是，如果系统所关注的时间段相对较短，例如，1 或 2 小时时间段的峰值需求，我们发现对很多系统(但肯定不是全部)来说到达速率在该区间上是相当稳定的，对该过程来说，泊松过程是这个区间的好模型(参见下面的定理 6.2 以及接下来的例 6.4)。

下面的定理是在 Cinlar(1975，第 74-76 页)中证明的，它解释了泊松过程名词的由来。

定理 6.1 如果$\{N(t), t\geqslant 0\}$是一个泊松过程，那么在长度为 s 的任意时间区间上到达的数目是参数为λs(其中 λ 为正实数)的泊松随机变量。也就是，

$$P[N(t+s)-N(t)=k]=\frac{e^{-\lambda s}(\lambda s)^k}{k!}, \quad k=0,1,2,\cdots \text{ 且 } t,s\geqslant 0$$

因此，$E[N(s)]=\lambda s$(参见第 6.2.3 小节)，且，特别是，$E[N(1)]=\lambda$。因此，λ 是长度为 1 的任意区间内到达的期望数目。我们称 λ 为过程的速率。

现在看到泊松过程的到达间隔时间是 IID 指数随机变量，参见 Cinlar(1975，第 78-80 页)。

定理 6.2 如果$\{N(t), t\geqslant 0\}$是速率为 λ 的泊松过程，那么它对应的到达间隔时间 A_1，A_2，…为 IID 指数随机变量，均值为 $1/\lambda$。

这个结果同上面的讨论一起解释了为什么发现一个有限制的时间段内的到达间隔时间往往近似为 IID 指数随机变量。例如，回忆一下，例 6.4 中便驾银行中车辆的到达间隔时间在 90min 时间段内近似为指数分布。

定理 6.2 的反命题也是正确的。也就是说，如果一个到达过程$\{N(t), t\geqslant 0\}$的到达间隔时间 A_1，A_2，…为 IID 指数随机变量，均值为 $1/\lambda$，那么$\{N(t), t\geqslant 0\}$是速率为 λ 的泊松过程[参见 Cinlar(1975，第 80 页]。

6.12.2 非平稳泊松过程

令 $\lambda(t)$为时刻 t 顾客到达系统的速率[关于 $\lambda(t)$的含义的某些见解见下面]。如果顾客以恒定速率 λ 的泊松过程到达系统，那么对于 $t\geqslant 0$，$\lambda(t)=\lambda$。然而，对于很多实际系统，$\lambda(t)$实际上是 t 的函数。例如，顾客到达快餐店的速率在中午忙的那个小时是大于下午中间的。高速公路上的车流量在早晚高峰期也要大一些。如果到达速率 $\lambda(t)$在实际上是随时间变化的，那么到达间隔时间 A_1，A_2，…分布并不相同；因此，使用 6.4 节～6.6 节中讨论的方法对 A_i 拟合单一概率分布是不合适的。在本节，讨论普遍使用的具有时变到达速率的到达过程的模型。

随机过程$\{N(t), t\geqslant 0\}$称为非平稳泊松过程，如果：

(1) 顾客每次到达一个。

(2) $N(t+s)-N(t)$独立于$\{N(u), 0\leqslant u\leqslant t\}$。

因此，对于一个非平稳泊松过程，顾客仍然必须每次到达一个，不相交区间的到达数目是独立的，但是现在到达速率 $\lambda(t)$允许是时间的函数。

令 $\Lambda(t)=E[N(t)]$，对于所有 $t\geqslant 0$。如果对于特定的 t 值 $\Lambda(t)$可微，我们正式地定义 $\lambda(t)$为：

$$\lambda(t)=\frac{d}{dt}\Lambda(t)$$

直观上看，在到达的期望数目大的区间 $\lambda(t)$将会很大。我们分别称 $\Lambda(t)$和 $\lambda(t)$为非平稳泊松过程的期望函数和速率函数。

下面的定理表明非平稳泊松过程在$(t, t+s]$区间内到达的数目是一个泊松随机变量，

其参数依赖于 t 和 s 两者。

定理 6.3　如果$\{N(t),\ t\geqslant 0\}$为具有连续的期望函数 $\Lambda(t)$的非平稳泊松过程，那么

$$P[N(t+s)-N(t)=k]=\frac{e^{-b(t,s)}\,[b(t,s)]^k}{k!},\quad k=0,1,2,\cdots \text{ 且 } t,s\geqslant 0$$

其中，$b(t,s)=\Lambda(t+s)-\Lambda(t)=\int_t^{t+s}\lambda(y)\mathrm{d}y$，对于$[t,\ t+s]$上所有但有限多个点，如果 $\mathrm{d}(\Lambda(t))/\mathrm{d}t$在$[t,\ t+s]$上是有界的且如果 $\mathrm{d}(\Lambda(t))/\mathrm{d}t$ 存在并连续，那么最后一个等式成立。

在还没有解决如何由感兴趣到达过程的一组观测数据来估计 $\lambda(t)$（或者 $\Lambda(t)$）的问题。下面的例子给出了一种启发式但实用的方法，在该例后简要地讨论其他方法。

例 6.26　为电子复印店开发了一个仿真模型，并对在 8 个不同的天的上午 11：00 和下午 1：00 之间的顾客到达的时间收集了数据。通过观察到达顾客的特征，感到非平稳泊松过程的性质(1)和(2)满足，此外，$\lambda(t)$在这两小时区间上是变化的。为了获得 $\lambda(t)$的估计，两小时区间分成下面的 12 个子区间：

$$[11:00,11:10),[11:10,11:20),\cdots,[12:40,12:50),[12:50,1:00)$$

对于每天，确定这些子区间每个的到达数。然后，对于每个子区间，计算出 8 天内子区间的到达平均数。这 12 个平均值是对应子区间到达期望数的估计。最后，对于每个子区间，用该子区间的平均到达数除以子区间长度 10min，以得到该子区间的到达速率的估计。图 6.57画出了估计的到达速率$\hat{\lambda}(t)$（用每分钟顾客的数目表示）的图形。注意，估计的到达速率在 2h 的时间段内变化明显。

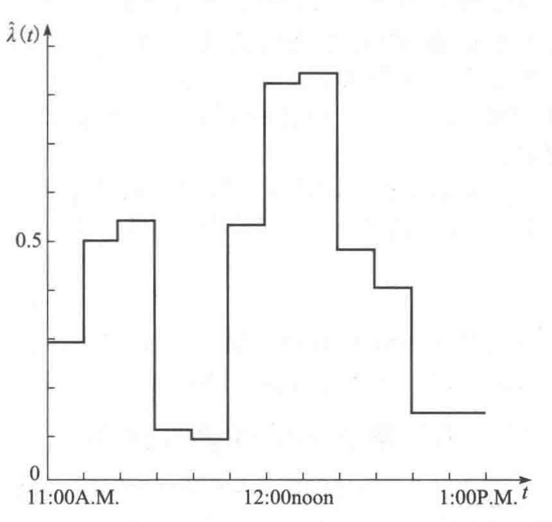


图 6.57　复印店上午 11：00 至下午 1：00 间到达过程每分钟的顾客数的估计速率函数$\hat{\lambda}(t)$图

有人可能很自然地要问，我们为何决定将区间长度定为 10min。实际上，我们用上面的方法将子区间分为 5、10、15min 长度计算过 $\lambda(t)$的估计值。基于子区间长度为 5min 的 $\lambda(t)$的估计值被拒绝了，因为感到$\hat{\lambda}(t)$的对应图形太不规则了，也就是说，子区间长度为 5min 太小。另一方面，也没有选择基于子区间长度为 15min 的$\lambda(t)$的估计，因为$\hat{\lambda}(t)$的对应图形看起来过于“平滑”，这意味着 $\lambda(t)$的真实性质的信息丢失了。一般来讲，这里选择子区间长度的问题同选择直方图区间宽度问题类似(参见第 6.4.2 小节)。◀

尽管例 6.26 中确定 $\lambda(t)$的分片常量法肯定是相当简单，并且相当灵活，但是它需要对常速率时间区间的边界和宽度进行一些主观判断。其他确定和估计 $\lambda(t)$[或者，另一个，$\Lambda(t)$]的方法已经开发出来，这里简要地介绍它们。

- 一些作者建议的确定估计的速率函数$\hat{\lambda}(t)$的方法是例 6.26 中做法的一般化，使用分段线性或者分段多项式形式。这包括 Kao 和 Chang(1988)，Lewis 和 Shedler (1976)，以及 Klein 和 Roberts(1984)。
- Leemis(1991)提出了一种直观、简单的非参数方法以确定期望函数 $\Lambda(t)$的分段线性估计，其中断点由叠加几个过程的实现中所观测的到达时间来确定。他证明了，随着所观测的实现的数目增加，估计值会以概率 1 收敛于真实的基本期望函数，而且他还推导出了一个真实 $\Lambda(t)$的置信带，它可以用于输入定义的灵敏度分析。由于估计的 $\Lambda(t)$是分段线性的，产生观测值(参见第 8.6.2 小节)简单有效。这种方法的一

般化在 Arkin 和 Leemis(2000)中给出。

- 确定和估计速率函数 $\lambda(t)$ 的一个不同方法就是，假设它具有一些特定的参数化(函数)形式，该形式在结构上足够通用，参数的个数也足够多，允许它较好地拟合所观测的数据。参数化形式应该考虑趋势和周期，也应该容纳参数估计的严格的统计方法，如最大似然方法或者最小二乘方法。这些函数，加上估计和产生的软件已由 Lee 等(1991)、Johnson 等(1994a、1994b)、Kuhl、Damerji 以及 Wilson(1997)、Kuhl 和 Wilson(2000)开发出来。
- Kuhl 和 Wilson(2001)给出了一种组合的参数/非参数方法以用于估计具有长运行趋势效果或者周期性效果，即呈现非三角特点的非平稳泊松过程的期望函数 $\Lambda(t)$。

6.12.3 批到达

对于某些实际系统，顾客是成批，或者说成组到达的，因此不满足泊松过程与非平稳泊松过程的性质(1)。例如，到达运动会或者食堂的人往往成批到达。现在考虑如何对这种到达过程建模。

现在令 $N(t)$ 为 t 时刻前已经到达的个体顾客的批数目。将本章前面讨论的方法应用到连续批的到达时间，可以开发过程 $\{N(t), t\geqslant 0\}$ 的模型。例如，如果批次的到达间隔时间近似为 IID 指数随机变量，$\{N(t), t\geqslant 0\}$ 就可以建模为泊松过程。之后，对连续批的大小拟合一个离散分布；批大小为正整数。因此，对于原始到达过程，可以假设顾客批次按 $\{N(t), t\geqslant 0\}$ 过程到达，每个批次的顾客数目为具有拟合的离散分布的随机变量。

上面的非正式讨论可以做得更加精确些。如果 $X(t)$ 为时刻 t 前到达的个体顾客总数，且如果 B_i 为第 $X(t)$ 批中顾客的数目，那么 $X(t)$ 表示为：

$$X(t) = \sum_{i=1}^{N(t)} B_i, \quad t \geqslant 0$$

如果假设 B_i 为 IID 随机变量，并且也独立于 $\{N(t), t\geqslant 0\}$，并且如果 $\{N(t), t\geqslant 0\}$ 是一个泊松过程，那么随机过程 $\{X(t), t\geqslant 0\}$ 就称为复合泊松过程。

6.13 不同数据集的同质性评测

有时候分析人员对一个随机现象独立地收集了 k 组观测值，他想知道这些数据集是否同质从而能合并。例如，有可能需要知道不同天收集的银行里顾客服务时间是否同质。如果它们是，来自这些不同天中的服务时间就可以合并，该组合的样本用于寻找其服务时间分布。否则，就需要多个服务时间分布。在本节中，我们讨论同质性的克鲁斯卡-沃尔斯(Kruskal-Wallis)假设检验。由于对数据的分布不做假设，这是非参数化检验。

假设我们有 k 个独立样本，可能有不同的大小，并且样本本身是独立的。对于 $i=1, 2, \cdots, k$，用 $X_{i1}, X_{i2}, \cdots, X_{in_i}$ 表示大小为 n_i 的第 i 个样本，并令 n 表示观测数据的总个数：

$$n = \sum_{i=1}^{k} n_i$$

然后我们要检验如下原假设：

H_0：所有的总体分布函数是相同的；

对应着备选假设：

H_1：至少其中一个总体生成的观测值趋向于比其他至少一个总体大。

为了构造一个克鲁斯卡-沃尔斯(K-W)统计，将 n 个观测值中最小的赋以秩 1，秩 2 对应于第二小，以此类推，直到 n 个观测值中最大的秩为 n。令 $R(X_{ij})$ 表示赋给 X_{ij} 的秩，且令 R_i 为赋给第 i 个样本的秩的和，也就是：

$$R_i = \sum_{j=1}^{n_i} R(X_{ij}), \quad i = 1,2,\cdots,k$$

那么，K-W 检验统计量 T 定义为：

$$T = \frac{12}{n(n+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(n+1)$$

如果 $T > \chi^2_{k-1,1-\alpha}$，那么我们在水平 α 上拒绝原假设 H_0，其中，$\chi^2_{k-1,1-\alpha}$ 为 $k-1$ 自由度 χ^2 分布的上 $1-\alpha$ 临界值。上面 T 的表达式假设没有两个观测值是相等的。如果不是这种情况，那么必须使用不同的 T 的表达式[参见 Conover(1999，第 288-290 页)]。

例 6.27 为了确定所需卸货设备的数目，开发了一个通宵空运服务的中心操作的仿真模型。该模型包括飞机到达可能早于或者迟于其计划到达时间的规定。有两个不同的进港航班号(每个对应于一个不同的出发城市)57 个不同天的实际到达时间数据可用(每个航班号每天到达一次，一周 5 天)。令 X_{ij} 为第 $j(j=1, 2, \cdots, 57)$ 天、航班号 $i(i=1, 2)$ 的计划到达时间减去实际到达时间(单位：分钟)。如果 $X_{ij}<0$，那么 i 号的航班在第 j 天迟到了。我们想做一个 K-W 检验以确定 X_{1j} 和 X_{2j} 是否同质，即两个航班号的到达模式是否相似。我们计算了 K-W 统计量并得到了 $T=4.317$，这大于临界值 $2.706=\chi^2_{1,0.90}$。因此，在 $\alpha=0.010$ 的水平上拒绝原假设，因此两个航班号的到达模式必须分开建模。观测到的差别在很大程度上是由于两个出发城市的不同天气造成的。◀

附录 6A　伽马分布和贝塔分布的 MLE 表

表 6.21　$\hat{\alpha}$ 作为 T 的函数，伽马分布

T	$\hat{\alpha}$	T	$\hat{\alpha}$	T	$\hat{\alpha}$	T	$\hat{\alpha}$
0.01	0.010	0.50	0.338	2.70	1.495	5.80	3.057
0.02	0.019	0.60	0.396	2.80	1.546	6.00	3.157
0.03	0.027	0.70	0.452	2.90	1.596	6.20	3.257
0.04	0.036	0.80	0.507	3.00	1.647	6.40	3.357
0.05	0.044	0.90	0.562	3.10	1.698	6.60	3.458
0.06	0.052	1.00	0.616	3.20	1.748	6.80	3.558
0.07	0.060	1.10	0.669	3.30	1.799	7.00	3.658
0.08	0.068	1.20	0.722	3.40	1.849	7.20	3.759
0.09	0.076	1.30	0.775	3.50	1.900	7.40	3.859
0.10	0.083	1.40	0.827	3.60	1.950	7.60	3.959
0.11	0.090	1.50	0.879	3.70	2.001	7.80	4.059
0.12	0.098	1.60	0.931	3.80	2.051	8.00	4.159
0.13	0.105	1.70	0.983	3.90	2.101	8.20	4.260
0.14	0.112	1.80	1.035	4.00	2.152	8.40	4.360
0.15	0.119	1.90	1.086	4.20	2.253	8.60	4.460
0.16	0.126	2.00	1.138	4.40	2.353	8.80	4.560
0.17	0.133	2.10	1.189	4.60	2.454	9.00	4.660
0.18	0.140	2.20	1.240	4.80	2.554	9.20	4.760
0.19	0.147	2.30	1.291	5.00	2.655	9.40	4.860
0.20	0.153	2.40	1.342	5.20	2.755	9.60	4.961
0.30	0.218	2.50	1.393	5.40	2.856	9.80	5.061
0.40	0.279	2.60	1.444	5.60	2.956	10.00	5.161

（续）

T	$\hat{\alpha}$	T	$\hat{\alpha}$	T	$\hat{\alpha}$	T	$\hat{\alpha}$
10.50	5.411	15.00	7.663	19.50	9.914	24.00	12.164
11.00	5.661	15.50	7.913	20.00	10.164	24.50	12.414
11.50	5.912	16.00	8.163	20.50	10.414	25.00	12.664
12.00	6.162	16.50	8.413	21.00	10.664	30.00	15.165
12.50	6.412	17.00	8.663	21.50	10.914	35.00	17.665
13.00	6.662	17.50	8.913	22.00	11.164	40.00	20.165
13.50	6.912	18.00	9.163	22.50	11.414	45.00	22.665
14.00	7.163	18.50	9.414	23.00	11.664	50.00	25.166
14.50	7.413	19.00	9.664	23.50	11.914		

表 6.22 $\hat{\alpha}_1$、$\hat{\alpha}_2$ 作为 G_1、G_2的函数，贝塔分布

如果 $G_1 \leqslant G_2$，使用第一列标记；如果 $G_2 \leqslant G_1$，使用第二列标记

G_1	G_2	$\hat{\alpha}_1$	$\hat{\alpha}_2$	G_1	G_2	$\hat{\alpha}_1$	$\hat{\alpha}_2$
G_2	G_1	$\widetilde{\alpha}_2$	$\widetilde{\alpha}_1$	G_2	G_1	$\widetilde{\alpha}_2$	$\widetilde{\alpha}_1$
0.01	0.01	0.112	0.112	0.05	0.40	0.259	0.494
0.01	0.05	0.126	0.157	0.05	0.45	0.271	0.560
0.01	0.10	0.135	0.192	0.05	0.50	0.284	0.640
0.01	0.15	0.141	0.223	0.05	0.55	0.299	0.739
0.01	0.20	0.147	0.254	0.05	0.60	0.317	0.867
0.01	0.25	0.152	0.285	0.05	0.65	0.338	1.037
0.01	0.30	0.157	0.318	0.05	0.70	0.366	1.280
0.01	0.35	0.163	0.354	0.05	0.75	0.403	1.655
0.01	0.40	0.168	0.395	0.05	0.80	0.461	2.305
0.01	0.45	0.173	0.441	0.05	0.85	0.566	3.682
0.01	0.50	0.179	0.495	0.05	0.90	0.849	8.130
0.01	0.55	0.185	0.559	0.05	0.94	2.898	45.901
0.01	0.60	0.192	0.639	0.10	0.10	0.245	0.245
0.01	0.65	0.200	0.741	0.10	0.15	0.262	0.291
0.01	0.70	0.210	0.877	0.10	0.20	0.278	0.337
0.01	0.75	0.221	1.072	0.10	0.25	0.294	0.386
0.01	0.80	0.237	1.376	0.10	0.30	0.310	0.441
0.01	0.85	0.259	1.920	0.10	0.35	0.327	0.503
0.01	0.90	0.299	3.162	0.10	0.40	0.345	0.576
0.01	0.95	0.407	8.232	0.10	0.45	0.365	0.663
0.01	0.98	0.850	42.126	0.10	0.50	0.389	0.770
0.05	0.05	0.180	0.180	0.10	0.55	0.417	0.909
0.05	0.10	0.195	0.223	0.10	0.60	0.451	1.093
0.05	0.15	0.207	0.263	0.10	0.65	0.497	1.356
0.05	0.20	0.217	0.302	0.10	0.70	0.560	1.756
0.05	0.25	0.228	0.343	0.10	0.75	0.660	2.443
0.05	0.30	0.238	0.387	0.10	0.80	0.846	3.864
0.05	0.35	0.248	0.437	0.10	0.85	1.374	8.277

（续）

$\frac{G_1}{G_2}$	$\frac{G_2}{G_1}$	$\frac{\hat{\alpha}_1}{\tilde{\alpha}_2}$	$\frac{\hat{\alpha}_2}{\tilde{\alpha}_1}$	$\frac{G_1}{G_2}$	$\frac{G_2}{G_1}$	$\frac{\hat{\alpha}_1}{\tilde{\alpha}_2}$	$\frac{\hat{\alpha}_2}{\tilde{\alpha}_1}$
0.10	0.89	5.406	44.239	0.25	0.45	0.724	1.007
0.15	0.15	0.314	0.314	0.25	0.50	0.822	1.254
0.15	0.20	0.335	0.367	0.25	0.55	0.962	1.624
0.15	0.25	0.357	0.424	0.25	0.60	1.186	2.243
0.15	0.30	0.380	0.489	0.25	0.65	1.620	3.486
0.15	0.35	0.405	0.563	0.25	0.70	2.889	7.230
0.15	0.40	0.432	0.653	0.25	0.74	12.905	37.229
0.15	0.45	0.464	0.762	0.30	0.30	0.647	0.647
0.15	0.50	0.502	0.903	0.30	0.35	0.717	0.777
0.15	0.55	0.550	1.090	0.30	0.40	0.804	0.947
0.15	0.60	0.612	1.353	0.30	0.45	0.920	1.182
0.15	0.65	0.701	1.752	0.30	0.50	1.086	1.532
0.15	0.70	0.842	2.429	0.30	0.55	1.352	2.115
0.15	0.75	1.111	3.810	0.30	0.60	1.869	3.280
0.15	0.80	1.884	8.026	0.30	0.65	3.387	6.779
0.15	0.84	7.908	42.014	0.30	0.69	15.402	34.780
0.20	0.20	0.395	0.395	0.35	0.35	0.879	0.879
0.20	0.25	0.424	0.461	0.35	0.40	1.013	1.101
0.20	0.30	0.456	0.537	0.35	0.45	1.205	1.430
0.20	0.35	0.491	0.626	0.35	0.50	1.514	1.975
0.20	0.40	0.531	0.735	0.35	0.55	2.115	3.060
0.20	0.45	0.579	0.873	0.35	0.60	3.883	6.313
0.20	0.50	0.640	1.057	0.35	0.64	17.897	32.315
0.20	0.55	0.720	1.314	0.40	0.40	1.320	1.320
0.20	0.60	0.834	1.701	0.40	0.45	1.673	1.827
0.20	0.65	1.016	2.352	0.40	0.50	2.358	2.832
0.20	0.70	1.367	3.669	0.40	0.55	4.376	5.837
0.20	0.75	2.388	7.654	0.40	0.59	20.391	29.841
0.20	0.79	10.407	39.649	0.45	0.45	2.597	2.597
0.25	0.25	0.500	0.500	0.45	0.50	4.867	5.354
0.25	0.30	0.543	0.588	0.45	0.54	22.882	27.359
0.25	0.35	0.592	0.695	0.49	0.49	12.620	12.620
0.25	0.40	0.651	0.830	0.49	0.50	24.873	25.371

习题

6.1　假设某人的工作是为在建飞机的右翼安装 98 个铆钉。如果随机变量 T 为一架飞机所需的总时间，那么 T 的近似分布是什么？

6.2　证明表 6.3 中韦布尔分布的注解 2。

6.3　证明表 6.3 中皮尔逊Ⅵ型分布的注解 2。

6.4　考虑一个 4 变量皮尔逊Ⅵ型分布，形状参数为 α_1 和 α_2，比例参数为 β，位置参数为 γ。如果 $\alpha_1=1$，$\gamma=\beta=c>0$，那么密度函数为：

$$f(x)=\alpha_2 x^{-(\alpha_2+1)}c^{\alpha_2},\quad x>c$$

这也是参数为 c 和 α_2 的帕雷托(Pareto)分布的密度函数，记作 $X\sim\text{Pareto}(c,\alpha_2)$。证明当且仅当 $Y=\ln X\sim\text{expo}(\ln c,1/\alpha_2)$，也就是位置参数为 $\ln c$、比例参数为 $1/\alpha_2$ 的指数分布时，$X\sim\text{Pareto}(c,\alpha_2)$。

6.5 对第 6.2.4 小节中由 $F(x)$ 给出的经验分布，讨论看似直观的定义 $F(X_{(i)})=i/n$，$i=1,2,\cdots,n$ 的优点。在这种情况下，对于 $0\leqslant x<X_{(1)}$，你会怎么定义 $F(x)$？

6.6 计算 6.2.4 小节中由 $F(x)$ 给出的经验分布的期望。

6.7 对于离散分布，证明直方图(第 6.4.2 小节)是未知质量函数的无偏估计，即证明，对于所有的 j，$E(h_j=p(x_j))$。提示：对于所有固定的 j，定义：

$$Y_j=\begin{cases}1, & X_i=x_j,\quad i=1,2,\cdots,n\\0, & \text{其他}\end{cases}$$

6.8 假设你观测的数据的直方图有多个局部众数(参见图 6.31)，但是不可能将数据分成具有不同概率分布的自然组来对每个组进行拟合。描述你对数据建模的其他方法。

6.9 对于参数为 p 的几何分布，解释为什么 MLE 的$\hat{p}=1/[\overline{X}(n)+1]$是直观的。

6.10 对于下面的每个分布，推导所列参数的 MLE 的公式。假设我们已经从讨论中的分布得到了 IID 数据 X_1，X_2，…，X_n。

(a) $U(a,b)$，b 的 MLE；

(b) $U(a,0)$，a 的 MLE；

(c) $U(a,b)$，a 和 b 的联合 MLE；

(d) $N(\mu,\sigma^2)$，μ 和 σ 的联合 MLE；

(e) $LN(\mu,\sigma^2)$，μ 和 σ 的联合 MLE；

(f) Bernoulli(p)，p 的 MLE；

(g) $DU(i,j)$，i 和 j 的联合 MLE；

(h) $B(t,p)$，假设 t 已知，求 p 的 MLE；

(i) negbin(s，p)，假设 s 已知，求 p 的 MLE；

(j) $U(\theta-0.5,\theta+0.5)$，求 θ 的 MLE。

6.11 对于参数为 λ 的泊松分布，对给定数据 X_1，X_2，…，X_n，推导 λ 近似百分之$100(1-\alpha)$置信区间(利用 MLE $\hat{\lambda}$的渐进正态性)。

6.12 考虑一个平移的指数分布(双参数)，对于 $\beta>0$ 以及任意实数 γ，具有以下密度函数：

$$f(x)=\begin{cases}\dfrac{1}{\beta}e^{-(x-\gamma)/\beta}, & x\geqslant\gamma\\0, & \text{其他}\end{cases}$$

已知样本 X_1，X_2，…，X_n 为该分布的 IID 随机变量，求联合 MLE 的$\hat{\gamma}$和$\hat{\beta}$的公式。提示：回忆一下，γ 不可能大于任何 X_i。

6.13 对频率比较，证明若拟合分布事实上为真实分布，则式(6.4)给出的 r_j 实际上是 n 个观测值中落在第 j 个区间内的期望比例。

6.14 对于 Q-Q 图，让经验分布函数$\widetilde{F}_n(x)$满足$\widetilde{F}_n(x_n)=1$，为什么会不方便？

6.15 当你试图为离散分布定义 Q-Q 图时会有什么困难呢？

6.16 假设真实分布 $F(x)$和拟合分布$\widetilde{F}_n(x)$相同，那么当样本大小 n 很大的时候，什么分布 $F(x)$的 Q-Q 图和 P-P 图本质上会是相同的？

6.17 如果拟合的分布事实上是真实分布，对 χ^2 检验来说，假设随机变量 M_i 为落在第 j 个区间$[a_{j-1},a_j)$内的 X_i 的个数。M_i 的分布是什么，均值呢？

6.18 对于 χ^2 检验，直观解释在某些情况下为什么 Kallenberg、Oosterhoff 和 Schriever(1985)发现了当 np_j 在尾部较小而不是完全相同的时候，能力较大？

6.19 令 $F_n(x)$为用于 K-S 检验的经验分布函数。证明对所有的 x，当 $n\to+\infty$时 $F_n(x)\to F(x)$(以概率 1)，其中 $F(x)$为真实基本分布函数。

6.20 使用 ExpertFit 的学生版，按照第 6.7 节中的步骤(也就是，数据求和、直方图、拟合分布并排序、密度直方图、分布函数差图、P-P 图、χ^2 检验、K-S 检验以及 A-D 检验)分析这些数据。对于水平为 $\alpha=0.05$ 进行检验。

a) 91 个实验室处理时间；

b) 1 000 个纸卷值(用需要用第二最好模型进行检验)；

c) 218 个邮局服务时间(用需要用第二最好模型进行检验)；

d) 一个 ATM 的 1 592 个时间；

e) 694 个机器维修时间；

f) 3 035 个麻醉后恢复时间。

6.21 假设表 6.24 中的数据(ExpertFit 中有)为某机器的维修时间(以分钟为单位)的独立观测值。使用 ExpertFit 的学生版按照第 6.7 节中的步骤(也就是，数据求和、直方图、拟合分布并排序、密度直方图、分布函数差图、P-P 图、χ^2 检验、K-S 检验，以及 A-D 检验)分析这些数据。用水平为 $\alpha=0.05$进行检验。

a) 369 个大学考试分数；

b) 在 19 世纪前十年，在普鲁士军队中每年被马踢死的 200 个。

6.22 假有两个机器维修时间数为 IID 随机变量，名字分别为 PROB622a 和 PROB622b(机器来自同一个销售商)，使用 ExpertFit 的学生版分析这些数据，进行 Kruskal-Wallis 检验(见 6.13 节)，用水平为 $\alpha=0.05$ 检验数据集是否是均匀分布。

6.23 对第 6.8 节中的位置参数估计$\widetilde{\gamma}$，证明当且仅当 $X_{(k)}=[X_{(1)}+X_{(2)}]/2$ 时，$\widetilde{\gamma}<X_{(1)}$。

6.24 令 $LN(\gamma,\mu,\sigma^2)$表示平移(三参数)对数正态分布，对于 $\sigma>0$ 以及任意的实数 γ 和 μ，密度函数为：

$$f(x)=\begin{cases}\dfrac{1}{(x-r)\sqrt{2\pi\sigma^2}}\exp\dfrac{-[\ln(x-\gamma)-\mu]^2}{2\sigma^2}, & x>\gamma\\ 0, & \text{其他}\end{cases}$$

因此，$LN(\mu,\sigma^2)$为原始的 $LN(\mu,\sigma^2)$分布。

(a) 校验，当且仅当 $X-\gamma\sim LN(\mu,\sigma^2)$时，$X\sim LN(\gamma,\mu,\sigma^2)$。

(b) 证明对于一个固定的已知 γ 值，$LN(\gamma,\mu,\sigma^2)$中 μ 和 σ 的 MLE 分别为：

$$\hat{\mu}=\frac{\sum_{i=1}^{n}\ln(X_i-\gamma)}{n},\hat{\sigma}=\left\{\frac{\sum_{i=1}^{n}[\ln(X_i-\gamma)-\hat{\mu}]^2}{n}\right\}^{1/2}$$

即只要将数据平移一个数量 γ，然后将它们作为(未平移)对数正态分布数据来对待。

6.25 对第 6.12.2 小节中的定理 6.3，直观解释为什么在区间$(t,t+s]$内到达的期望数量$b(t,s)$应等于 $\int_t^{t+s}\lambda(y)\mathrm{d}y$。

6.26 通过下面的步骤(a)到步骤(c)，给出在连续情况下(参见第 6.5 节)定义 MLE 的直观原因。同以前一样，观测数据 $X_1,X_2,\cdots,X_n$ 为随机变量 X 的 IID 实现，密度函数为 f_θ。记住 X_i 已经观测出来，因此应当看做固定的数值，而不是随机变量。

(a) 令 ε 为很小(但严格为正)的实数，并定义短语“在 X_i 附近获得一个 X 值”为事件$\{X_i-\varepsilon<X<X_i+\varepsilon\}$。使用微积分中的中值定理证明，对于任意的 $i=1,2,\cdots,n$，P(在 X_i 附近获得一个 X 值)$\approx 2\varepsilon f_\theta(X_i)$。

(b) 定义短语“在观测数据附近得到 X 的 n 个 IID 值的一个样本”为事件(在 X_1 附近获得一个 X 值，在 X_2 附近获得一个 X 值，…，在 X_n 附近获得一个 X 值)。证明 P(在观测数据附近得到 X 的 n 个 IID 值的一个样本)$\approx(3\varepsilon)^n f_\theta(X_1)f_\theta(X_2)\cdots f_\theta(X_n)$，注意这正比于最大似然函数$L(\theta)$。

(c) 证明 MLE 的$\hat{\theta}$是在观测数据附近得到 X 的 n 个 IID 值的一个样本的近似概率的最大化的 θ 的值，在这个意义上“最好的解释了”实际观测的数据。

6.27 为什么在表 6.2 中队列的平均延误近似等于对应的队列中平均数?

6.28 证明第 6.11 节中等式(6.17)是正确的。

6.29 开发牛顿法通用迭代公式以用于第 6.11 节[参见式(6.22)]中韦布尔分布形状参数 α 估计。公式应具有以下形式：

$$\widetilde{\alpha}_{k+1}=\widetilde{\alpha}_k-\frac{f(\widetilde{\alpha}_k)}{f'(\widetilde{\alpha}_k)}$$

其中，f'表示 f 的导数。

6.30 在缺少数据的情况下(第 6.11 节)，说明如何基于 a、m 和 x_q 的主观估计来确定一个三角分布。

6.31 在缺少数据的情况下(第 6.11 节)，说明如何基于 a，b 和 μ 代替 a，b 和 m 确定一个三角分布。

第 7 章 随机数发生器

7.1 引言

从某种意义上讲，固有随机成分的任何系统或者过程的仿真需要产生或者得到随机数的方法。例如，第 1 章与第 2 章的排队模型和库存模型到达间隔时间、服务时间、需求量等，它们需要从某个特定的分布，譬如正态分布或者厄兰分布中“获得”。在本章和下一章将讨论如何方便而有效地从所需概率分布产生随机数以用于执行仿真模型。在数学的概率论中，随机变量定义为满足某种条件的函数，因此“产生随机变量”这种说法严格来讲是不正确的，为了避免这种说法，我们将采用更为准确的术语即“产生随机变数”的说法。

本章致力于讨论由[0，1]区间上的均匀分布产生随机变数的方法，这种分布在第 6 章记作 $U[0, 1]$。从 $U[0, 1]$分布中生成的随机变数称为随机数。虽然均匀分布是所有连续分布中最简单的一种，能得到这种独立的随机数是特别重要的。$U[0, 1]$分布之所以具有如此高的地位，是因为所有其他分布(正态分布、伽马分布、二项分布等)的随机变数以及各种随机过程(如非平稳泊松过程)的实现都可以通过按所要求的分布或过程确定的方法对 IID 随机数进行变换的方法来得到。本章将讨论得到独立的随机数的方法，下一章讨论将随机数进行变换的方法以得到其他分布的变数或实现各种随机过程。

产生随机数的方法学有着漫长而有趣的历史，参见 Hull 和 Dobell(1962)，Morgan(1984，第 51-56 页)以及 Dudewicz(1975)的生动介绍。最早期的方法本质上是由手工进行的，如抽签(《马太福音》27：35)、掷骰子、发牌，以及从“混合均匀的罐子”中取出标有号码的小球。正如很多生于 20 世纪 60 年代末 70 年代初的美国男性所熟知的那样，很多彩票开奖依然是采用这种方式来操作的。在 20 世纪初期，出于对随机数的浓厚兴趣，许多统计学家加入了赌徒的行列，发明了一些机械化的设备，以便更快速地产生随机数；到了 1930 年代后期，Kendall 和 Babington-Smith(1938)利用高速转盘创建了一个含有 100 000 个随机数字的表。又过了一些时间，开发出了基于随机脉冲真空管的电子电路，以每秒 50 个的速度产生随机数字。英国邮政总局曾经使用这样的一个随机数机——电子随机数指示设备(electronic random number indicator equipment，ERNIE)——来决定保险储蓄债券(premium saving bond)彩票的获奖者[见 Thomson(1959)]。兰德公司(Rand Corporation)(1955)使用过另一个电子设备来产生有 100 万个随机数字的表。人们还发明了很多其他的方法，例如从电话号码簿或者人口普查报告中，或者利用 π 的 100 000个十进制位数字的展开式“随机地”选取数字。近年来，有人对建造和测试物理随机数“机器”产生了兴趣；例如，Miyatake 等(1983)介绍了一种基于计算伽马射线数的设备。

随着计算机(和仿真)的越来越广泛地应用，对与用计算机工作的办法兼容的随机数生成的方法得到了越来越多的关注。一种可行的方式就是将电子随机数机，例如 ERNIE，直接挂到在计算机上。但是这有一些缺点，主要是我们不能严格地再生先前生成的随机数流(关于需要再生随机数流的原因将在本节后面的内容中讨论)。另一种可行的方式恐怕是从表格中读取数据，例如兰德公司表格，但是这样做要么需要很大的内存，要么需要很多时间以用于相对较慢的输入操作(而且，对现代大规模仿真来说，使用远多于上百万个随

机数并不罕见，而且每个随机数需要若干位随机数字)。因此，在 1940 年和 1950 年代的研究转向数字或算术的方法来产生“随机”数。这些方法是序贯式的，每个新数按照一个固定数学公式由前面的一个或者几个数来确定。第一个这样的算术发生器是著名的平方取中法，由 Neumann 和 Metropolis 在 1940 年提出的，该方法的例子如下。

例 7.1 从一个 4 位正整数 Z_0 出发，取它的平方以得到一个不超过 8 位的整数；如果不够 8 位，则在其左侧补零使它刚好 8 位。取出该 8 位数的中间 4 位作为下一个 4 位正整数 Z_1。在 Z_1 的左侧加上十进制小数点，就得到了第一个“$U[0, 1]$随机数”，U_1。然后，令 Z_2 为 Z_1^2 的中间 4 位，并令 U_2 为 Z_2 的左侧加上十进制小数点后得到的小数，以此类推。表 7.1 列出了取 $Z_0=7\,182$ 时，得到的前几个 Z_i 和 U_i(初始 4 位正整数 Z_0 是自然对数的底数 e 的十进制小数部分的前 4 位)。

表 7.1　平方取中法

i	Z_i	U_i	Z_i^2
0	7 182	—	51 581 124
1	5 811	0.581 1	33 767 721
2	7 677	0.767 7	58 936 329
3	9 363	0.936 3	87 665 769
4	6 657	0.665 7	44 315 649
5	3 156	0.315 6	09 960 336
⋮	⋮	⋮	⋮

◀

从直觉上来看，平方取中法从一个数得到下一个数似乎提供了一个好的不规则性，从而可能认为这样一个非常具有偶然性的规则会提供一种相当好的方法来产生随机数。实际上，这种方法做得并不非常好。一个非常严重的问题(除了别的外)是，这种方法有很强的非常快速地退化到 0 的倾向，此后不再发生变化(例如，表 7.1 继续推进若干步，或者取 $Z_0=1\,009$，即兰德公司表格中的最初 4 个数字)。这表明了这样一种危险，即认为通过制定一些奇怪的规则从一个数得到另一个数总会得到一个好的随机数发生器。

从不可预测的意义上讲，平方取中法一个更基本的缺陷是它完全不是“随机”的。的确，如果我们已知一个数，那么下一个数是完全确定的，因为得到它的规则是固定的；实际上，一旦指定了 Z_0，Z_i 和 U_i 的整个序列就确定了。所有算术数发生器(在本章剩余部分我们讨论的都属于这一类)都存在这个缺陷，在争论这个问题时，人们往往很快就会开始迷惑真正的随机数的真实本质是究竟什么(有时算术随机数发生器称为伪随机，一个不好听的术语，我们不采用，虽然它可能更为准确)。的确，约翰·冯·诺伊曼(John von Neumann(1951))曾经说过下面一段被广泛引用的话：

> “考虑用算术方法产生随机数的任何人，当然是都是处于荒唐的状态。因为，正如人们多次指出的，根本没有随机数——只有产生随机数的方法，但严格的算术程序当然不是这种方法……我们在这里只不过是用“烹饪食谱”得到一系列数字。”

但是，很少有人说，诺伊曼在同一段中进一步不那么沮丧地说，这些“食谱”

> “……可能……无法得到证明，但是可以从其结果来进行判断。应该对采用某种给定食谱生成的数字进行统计研究，但是彻底的检验是不切实际的，如果这些数字在一个问题中表现令人满意，那么他们通常对同类的其他问题似乎也是成功的。”

这个更偏重实际应用的观点也被 Lehmer(1951)所认可。Lehmer 开发了迄今为止恐怕是使用最为广泛的一类随机数的生成方法，他的看法如下：

> "……算术随机数发生器的构想被视为是一个体现序列的构想的含糊概念，序列中的每一项对未知初始值来说是不可预测的，且其数字能够通过一定数量的惯例检验的，这些检验由统计学家并在某种程度上取决于该序列的应用进行的。"

Ripley(1987，第 19 页)在公理化意义下给出了"随机性"更为正式的定义；Niederreiter(1978)指出统计学的随机性也许并不是所希望的，所生成的数字的其他特性，如点的分布的"均匀性"在某些应用中更为重要，例如 Monte Carlo 积分。很多学者认为，经过精心的设计，算术发生器产生的数，它们通过一系列的统计学检验(见第 7.4 节)，能够表现出由 $U[0, 1]$分布得到的独立性，我们同意这样的观点。这是"随机数"的一个很有用的定义，我们采用该定义。

一个"好的"算术随机数发生器应当具备以下属性：

(1) 所产生的数字应当呈现出在[0, 1]区间上是均匀分布的，并且相互之间不应存在相关性；否则仿真的结果将会完全无效。

(2) 从实用的角度来说，我们自然希望发生器具有速度快，并且不需要过大的存储空间。

(3) 我们要能严格地重复生成一个给定的随机数流，原因至少有二。首先，这有时能使得调试或校验计算机程序更容易；更重要的，我们可能需要使用一组同样的随机数仿真不同的系统以便得到更为精确的比较，11.2 节将详细讨论这一点。

(4) 发生器应当备有易于产生分开的随机数流。一个随机数流只是发生器所产生的所有随机数的一部分，一个随机数流的开始是前一个随机数流的结束。可以将不同的随机数流看做分开的且相互独立的(假设没有用尽整个流，整个长度一般选得非常大)。因此，用户可以把某个特定的流"用于"仿真中的某个特定的随机因素。例如，在 2.4 节的单服务台排队模型中就是这样做的，其中流 1 用于产生到达间隔时间，而流 2 用于产生服务时间。针对不同的目的采用不同的流有利于再现和比较仿真结果。虽然这个概念直观上是显见的，仍有支持它的概率基础，正如 11.2 节中讨论的那样。多个流可用的其他优点在第 11 章中的其他部分讨论。对一个发生器而言，如果有一种有效方法从第 i 个随机数跳到第 $(i+k)$个随机数，k 为很大的数，则该发生器具有建立不同流的能力。

(5) 我们希望发生器是方便的，即对所有标准的编译器和计算机来说，产生相同的随机数序列(至少在机器所能达到的精度上相同)。

大部分常用的随机数发生器是相当快的，需要非常小的存储，而且能够很容易地产生指定的随机数序列，从而上述第(2)、(3)点几乎普遍满足。而且，大部分发生器，尤其是现代仿真软件包中所包括的那些发生器，现在都以某种方式实现了多流，满足第(4)点要求。不幸的是，也很多发生器未满足上述第(1)点的均匀性和独立性准则，如果人们希望得到正确的仿真结果，这是绝对必须满足的。Sawitzki(1985)的精品文章给出了大量的统计上不可接受的发生器。Park 和 Miller(1998)以及 L'Ecuyer(2001)报告了公开出版的发生器表现出了非常差的性能的几个例子，包括有一个只能不断重复同一个"随机数"。

7.2 节将讨论最为常见的一类发生器，而 7.3 节讨论一些其他方法。7.4 节讨论人们如何能对所希望的统计特性来检验一个给定随机数生成器。最后，附录 7A 和 7B 用 C 语言给出两个随机数发生器的可移植的计算机代码。第一个是第 1、2 章中用到的随机数发生器，第二个被认为具有较好的统计学属性，并建议用于实际应用。

随机数产生这个主题是一个非常复杂的主题，一方面涉及诸如抽象代数论和数论这样不同的学科，一方面包括，另一方面，它包括系统编程和计算机硬件工程。关于随机数发生器的一般文献有下列书籍：Fishman(1996，2001，2006)、Gentle(2003)、Knuth(1998)和 Tezuka(1995)，以及 L'Ecuyer(2006，2012)中的相关章节。

7.2 线性同余发生器

目前使用的许多随机数发生器是线性同余发生器(linear congruential generator, LCG)，这是由 Lehmer(1951)提出的。整数 Z_1，Z_2，…序列由下面的迭代公式定义：

$$Z_i = (aZ_{i-1} + c) \quad (\text{对 } m \text{ 取模}) \tag{7.1}$$

其中，m(模数)，a(乘子)，c(增量)以及 Z_0(种子或称为初始值)都是非负整数。因此，式(7.1)表示，用 $aZ_{i-1}+c$ 除以 m，并令 Z_i 为该除法的余数。从而，$0 \leqslant Z_i \leqslant m-1$，为了得到[0，1]区间上的随机数 $U_i(i=1, 2, \cdots)$，令 $U_i = Z_i/m$。虽然由于不同的计算机和编译器在处理浮点运算时的操作方式不同，应该注意 Z_i 被 m 除的精度性质。但是，将着重关心 Z_i 的大部分。除了非负性，整数 m、a、c 以及 Z_0 还应该满足 $0<m$，$a<m$，$c<m$，$Z_0<m$。

直接地，线性同余发生器就会有两个缺点。第一个缺点是所有(伪)随机数发生器共同的，即公式(7.1)定义的 Z_i 根本不是真正的随机数。事实上，通过数学推导，人们可以证明，对于 $i=1, 2, \cdots$，有：

$$Z_i = \left[a^i Z_0 + \frac{a(a^i - 1)}{a-1}\right] \quad (\text{对 } m \text{ 取模})$$

从而每一个 Z_i 由 m、a、c 和 Z_0 完全确定。然而，通过仔细选择这四个参数，我们努力使 Z_i 的特性保证相应的 U_i 在各种统计检验时都呈现出是独立同分布 $U(0, 1)$ 随机变数(见第7.4 节)。

线性同余发生器的第二个缺点也许是，U_i 的取值只能是有理值 0，$1/m$，$2/m$，…，$(m-1)/m$，事实上，U_i 实际只会取其中的一部分，这依赖于常数 m、a、c 和 Z_0 的规定，以及用 m 浮点除的性质。这样，U_i 不可能取到如 $0.1/m \sim 0.9/m$ 之间的数，虽然它应该以概率 $0.8/m>0$ 发生。通常选取很大的值，如 10^9 甚至更大作为模 m，这时，U_i 能落在在[0，1]上的点就十分密集；对于 $m \geqslant 10^9$，可能的取值至少有 10 亿个。

例 7.2 考虑由 $m=16$、$a=5$、$c=3$ 和 $Z_0=7$ 定义的线性同余发生器。表 7.2 给出了 $i=1, 2, \cdots, 19$ 时的 Z_i 和 U_i(保留到十进制小数点后 3 位)。注意，$Z_{17}=Z_1=16$，$Z_{18}=Z_2=1$，等等。也就是说，从 $i=17 \sim 32$，我们将得到与从 $i=1 \sim 16$ 时所得到的完全相同的 Z_i(以及 U_i)，并且顺序也完全一样(我们不会真的向任何人建议使用该发生器，因为 m 是如此之小，它只说明 LCG 的算法)。

表 7.2　LCG $Z_i=(5Z_{i-1}+3)$(对 16 取模)，$Z_0=7$

i	Z_i	U_i	i	Z_i	U_i	i	Z_i	U_i	i	Z_i	U_i
0	7	—	5	10	0.625	10	9	0.563	15	4	0.250
1	6	0.375	6	5	0.313	11	0	0.000	16	7	0.438
2	1	0.063	7	12	0.750	12	3	0.188	17	6	0.375
3	8	0.500	8	15	0.938	13	2	0.125	18	1	0.063
4	11	0.688	9	14	0.875	14	13	0.813	19	8	0.500

◀

例 7.2 中的 LCG 的“循环”特性是必然的。根据式(7.1)的定义，Z_i 一旦取得了以前出现过的值，就生成与之前完全一样的序列。这个循环无限地自身重复。该循环的长度称为发生器的周期。对于 LCG，Z_i 仅仅与前一个整数 Z_{i-1} 有关，并且因为 $0 \leqslant Z_i \leqslant m-1$，显然其周期至多为 m；如果事实上它就是 m，这该 LCG 称其为满周期(例 7.2 中的 LCG 就是满周期的)。显然，如果一个发生器是满周期的，不论初始种子 Z_0 取 $\{0, 1, \cdots, m-1\}$ 中的任何值都将以某种顺序产生整个循环。然而，如果发生器的周期小于满周期，循环的长度事实上取决于所选 Z_0 的特定值，在这种情况下我们实际应当指的是该发生器的种子的

周期。

因为大规模系统仿真项目可能使用几百万个随机数，显然希望具有长周期的 LCG。进一步，最好是满周期，因为我们就能保证在每个循环中 $0\sim m-1$ 之间的每个整数将恰好出现一次，这会有助于 U_i 的均匀性(但是，即使是满周期的 LCG，按一个循环的段来说，也可能呈现出非均匀性。例如，假如我们只生成 $m/2$ 个连续的 Z_i，它们在可能的取值 0，1，…，$m-1$ 序列中有可能留下大间隙)。所以，了解如何选择参数 m、a 和 c 使得相应的 LCG 具有满周期是有用的。下面的定理给出了这种特征描述，是由 Hull 和 Dobell(1962)证明的。

定理 7.1 由式(7.1)定义的 LCG 具有全周期，当且仅当下列三个条件成立：

(a) m 和 c 能够同时被整除的正整数只有 1。

(b) 如果 q 为整除 m 的素数(只能被 1 和其自身整除)，则 q 能整除 $a-1$。

(c) 如果 4 整除 m，则 4 整除 $a-1$。

定理 7.1 中的条件(a)通常表述为“m 与 c 互质。”

具有满周期(或者至少是长周期)正是好的 LCG 的一个所希望的性质。正如在 7.1 节中指出的，希望好的统计特性(如明显的独立性)、计算和存储效率、再生性、不同流的实现性，以及可移植性(见第 7.2.2 小节)。再生性简单，因为我们只需记住所用的初始种子 Z_0，再使用该值对发生器初始化就得到完全相同的 U_i 序列。而且，通过保存之前得到的最后一个 Z_i，并用它作为新的种子，我们能够很容易地在该序列的任何点继续产生 Z_i；这是得到不重叠的“独立”随机数序列的常用方法。

在 LCG 中，创建流的典型的做法是，对每个流指定其初始种子。例如，假设我们需要一组长度为 1 000 000 的流，那么取第一个流的 Z_0 为某个值，然后将 $Z_{1\,000\,000}$ 作为第二个流的种子，$Z_{2\,000\,000}$ 作为第三个流的种子，以此类推。所以，我们看到这些流实际上是所产生的单一随机数序列中不重叠的相邻子序列。在上面的例子中，如果一个流所用随机数超过 1 000 000 个，就会侵占下一个流的开头，而这些随机数可能已经用于某些地方，导致出现不希望的相关性。

在本节的剩余部分，将讨论如何选择参数以得到一个好的 LCG，并辨别出一些还在使用的性能差的 LCG。由于定理 7.1 中的条件(a)，$c>0$(称为混合 LCG)与 $c=0$(称为乘 LCG)将会有不同表现。

7.2.1 混合发生器

对于 $c>0$，定理 7.1 的条件(a)有可能满足，从而有可能得到满周期 m，正如现在讨论的。为了 U_i 有大周期且在[0，1]区间上有高密度，希望 m 要大。另外，早期的计算机仿真中，那时计算机相对较慢，在式(7.1)中除以 m 以得到余数是一个相对较慢的算术运算，所以希望避免必须显式地做这种除法。考虑到这些因素，好的选择就是取 $m=2^b$，其中，b 是所用计算机中一个字的位数(二进制数位)，以便于实际数据的存储。例如，大多数计算机和编译器具有 32 位字长，最左边的位是符号位，所以 $b=31$，且 $m=2^{31}>2.1$ 亿。而且，取 $m=2^b$ 可使在大多数计算机上通过整数溢出的优点来避免显式进行 m 的除法。能表示的最大整数是 2^b-1，任何力图存储一个更大的整数 W(譬如说，位数 $h>b$)，就会导致该超长的整数丢失其左边 $h-b$ 个二进制位，这时剩下的 b 位数刚好等于 $W(\mathrm{mod}\ 2^b)$。

采用 $m=2^b$ 这种选择，定理 7.1 表明，如果取 c 为奇数，$a-1$ 能够被 4 整除，将得到满周期。Z_0 可以是 $0\sim m-1$ 之间的任意整数而不影响周期。但是，在第 7.2 节的剩余部分，我们将着重于乘 LCG，因为它们的应用广泛得多。

7.2.2 乘法发生器

乘 LCG 的优点在于不需要加上参数 c，但这类发生器不可能有满周期，因为定理 7.1

中的条件(a)不可能满足(因为，例如，当 m 为正数，m 和 c 能够同时被整除的数是 0)。但是，正如我们将会看到的那样，如果仔细地选择 m 和 a，有可能得到周期 $m-1$。

与混合发生器一样，取 $m=2^b$，计算效率仍然高且避免显式除。但是，能够证明[例如，参见 Knuth(1998a，第 20 页)]，在这种情况下其周期最多为 2^{b-2}，也就是说，Z_i 所能取得数值的个数仅仅为 $0\sim m-1$ 之间所有整数个数的四分之一(实际上，如果 Z_0 为奇数，对于某个 $k=0, 1, \cdots, a$ 可以表示成 $8k+3$ 或者 $8k+5$ 这样的形式，则周期是 2^{b-2})。而且，一般无法得知这 $m/4$ 个整数将会落在什么地方；即所得到的 Z_i 中可能存在无法接受的大间隙。另外，如果选择 a 具有 2^l+j 这样的形式(从而，Z_{i-1} 与 a 相乘可以用移位操作和加 j 运算代替)，将会引入较差的统计特性。普遍熟悉的发生器 RANDU 就是这样的形式($m=2^{31}$，$a=2^{16}+3=65\,539$，$c=0$)，而且已经证明，这个发生器具有不尽如人意的统计特性(参见 7.4 节)。即使不选 $a=2^l+j$，在乘 LCG 使用 $m=2^b$ 恐怕不是一个好选择，就是因为其 $m/4$ 的短周期且导致有可能出现间隙。

由于在乘 LCG 中选 $m=2^b$ 带来的这些困难，人们将注意力放在寻找其他方法来规定 m 值。Hutchinson(1966)给出这样的方法，该方法已经证明是相当成功的，Hutchinson 将该想法归功于 Lehmer。该方法不是令 $m=2^b$，而建议 m 取小于 2^b 的最大素数。例如，在 $b=31$ 的情形，小于 2^{31} 的最大素数非常容易得到为 $2^{31}-1=2\,147\,483\,647$。现在，对于素数 m，可以证明，如果 a 为模 m 的素元，则周期为 $m-1$，即使 a^l-1 能被 m 整除的最小整数是 $l=m-1$，参见 Knuth(1998a，第 20 页)。对于以这种方法选择的 m 和 a，我们得到每个整数 $1, 2, \cdots, m-1$ 在每个循环内正好出现一次，所以 Z_0 可以是 $1\sim m-1$ 的任意整数，得到的周期仍是 $m-1$。这称为素数取模乘 LCG(PMMLCG)。

与 PMMLCG 有关的两个问题马上产生了：①如何得到模 m 的素元？虽然 Knuth(1998a，第 20-21 页)给出了某些特点，从计算角度来看，该任务是相当复杂的。我们将通过讨论下面两个广泛使用的 PMMLCG 在本质上解决这一点。②由于不选 $m=2^b$，再也不能直接使用整数溢出机制来达到取 m 模除法的效果。这种情况下，避免显式除法的方法也使用一类溢出，是由 Payne、Rabung 和 Bogyo(1969)给出的，并称为仿真除。下面讨论的 Marse 和 Robert 的可移植的发生器使用仿真除。

对于 PMMLCG 来说，重要的是取模 $m^*=2^{31}-1$ 的素元的好乘子 a，从而得到周期为 m^*-1。在一组重要的文献中，Fishman 和 Moore(1982，1986)对取模 m^* 的素元的所有乘子 a 进行了评价，数目约为 534 000 000 个。他们使用了实验和理论两种检验方法(参见下面第 7.4 节)，而且他们按照一系列相当严格的准则找出了一些性能较好的乘子。

对于模数 m^*，已经得到广泛使用的 a 的两个特定乘数值为 $a_1=7^5=16\,807$ 和 $a_2=630\,360\,016$，这两个都是取模 m^* 的素元[然而，Fishman 和 Moore 并没有发现哪个 a 值是其中最好的(参见第 7.4.2 小节)]。乘子 a_1 原是 Lewis，Goodman 和 Miller(1969)建议的，Schrage(1979)采用仿真除法在一个聪明的 FORTRAN 实现中使用了 a_1。Schrage 的代码的重要性在于，他在当时提供了一个相当好且便于移植的随机数发生器。

乘子 a_2 原是 Payne，Rabung 和 Bogyo(1969)建议的，Fishman 和 Moore 发现其得出的统计性能优于 a_1(参见第 7.4.2 小节)。Marse 和 Roberts(1983)提供了该乘子的具有很强移植性的 FORTRAN 例程，在附录 7A 中给出了该发生器的 C 版本。在本书第 1、2 章中的所有例子我们使用的都是这个发生器，同时，它也被放到第 2 章的 Simlib 软件包中。

具有 $m=m^*=2^{31}-1$ 且 $a=a_2=630\,360\,016$ 的 PMMLCG 对某些应用可提供可接受的结果，尤其是如果所需随机数的数量不是很大的话。然而许多专家[例如，L'Ecuyer，Simard，Chen 和 Kelton(2002)以及 Gentle(2010，第 21 页)]建议，模在 2^{31} 左右的 LCG 不应用作通用仿真软件包中的随机数发生器(如离散事件仿真)。原因在于，不仅是一个周期的随机数在很多计算机上往往在几分钟内就被用尽；而且，更重要的是，这些随机数相

对较差的统计性会使得仿真结果偏离样本长度，后者较发生器周期要小得多。例如，L'Ecuyer和 Simard(2001)发现，当样本中的观测数量约为发生器周期长度立方根的 8 倍时，与从 $U[0,1]$分布所得到的独立观测的样本中所期望的相比，模为 m^*，乘子为 a_1 或者 a_2 的 PMMLCG 表现出一定差异。这样，这些发生器的“可靠”周期实际上大约只有 10 000[另参见 L'Ecuyer 等(2000)]。

如果希望随机数发生器具有更大的周期和更好的统计特性，那么就应该考虑 L'Ecuyer 的组合多重递归发生器或者 Mersenne Twister 发生器(分别参见第 7.3.2 小节和第 7.3.3 小节)。

7.3 其他类型的发生器

虽然 LCG 可能是使用最广泛和最易于理解的一类随机数发生器，但是还有很多其他类型(7.1 节中的平方取中法，但是这种方法并不值得推荐)。开发这些其他类型发生器的主要目的是要得到更长的周期和更好的统计特性。这一节的讨论并不意味着给出所有类型的发生器的详细情况，而只是说明某些主要的可替代 LCG 的发生器。

7.3.1 更一般的同余

LCG 可以看做是由下面这种发生器定义的一种特殊情况：

$$Z_i = g(Z_{i-1}, Z_{i-2}, \cdots) \quad (\text{对 } m \text{ 取模}) \tag{7.2}$$

其中，g 是之前的 Z_j 的一个固定的确定性函数。

与 LCG 一样，由式(7.2)定义的 Z_i 都位于 $0 \sim m-1$ 之间，$U_i = Z_i/m$ 就得到 $U(0,1)$ 随机数[对于 LCG，函数 g 当然满足 $g(Z_{i-1}, Z_{i-2}, \cdots) = aZ_{i-1} + c$]。这里简要讨论这些类型的发生器的少数几个，要了解更详细的讨论，读者可以参考 Knuth(1998a，第 26-36 页)以及 L'Ecuyer(2012)。

LCG 的一个显而易见的推广应是令 $g(Z_{i-1}, Z_{i-2}, \cdots) = a'Z_{i-1}^2 + aZ_{i-1} + c$，它得到二次同余发生器。一种特殊情形是 $a' = a = 1$，$c = 0$，且 m 为 2 的幂；虽然这个特殊的发生器所得到的结果很接近平方取中法(参见第 7.1 节)，但它有更好的统计性能。因为 Z_i 仍然仅仅取决于 Z_{i-1}(而与更早的 Z_j 无关)，并且因为 $0 \leqslant Z_i \leqslant m-1$，所以与 LCG 一样，二次同余发生器的周期至多是 m。

函数 $g()$的另一种选择是保持线性，但是使用更早的 Z_j，这样得到的发生器称为多重递归发生器(multiple recursive generator，MRG)，其定义为：

$$g(Z_{i-1}, Z_{i-2}, \cdots) = a_1 Z_{i-1} + a_2 Z_{i-2} + \cdots + a_q Z_{i-q} \tag{7.3}$$

其中，a_1，a_2，$\cdots$，a_q 为常数。

这时，如果参数选择恰当，周期有可能大到 $m^q - 1$[见 Knuth(1998a，第 29-30 页)]。L'Ecuyer，Blouin 和 Couture(1993)研究了这类发生器，引入了广义谱检验对它们进行评价(见 7.4.2 小节)；作者还找到几个特别的性能较好的这类发生器，并给出了可移植的实现。以式(7.3)形式用于式(7.2)的 $g()$的其他发生器，人们的关注集中在将 $g()$定义为 $Z_{i-1} + Z_{i-q}$的情况，它包括古老的斐博拉西发生器：

$$Z_i = (Z_{i-1} + Z_{i-2}) \quad (\text{对 } m \text{ 取模})$$

这种发生器的周期会超过 m，但从统计学的角度来看，是完全不可接受的，参见习题 7.12。

Haas(1987)提出了沿着不同的路线来实现 LCG 的广义化，他建议在式(7.1)的基本 LCG 中，在产生每个新的 Z_i 之前，按照同余公式，我们对乘子 a 和增量 c 都进行变化。这类发生器的统计检验(参见第 7.4.1 小节)结果是令人欣慰的，他的分析表明我们的确能够得到非常大的周期，例如，在一个例子中达到 800 万亿。

7.3.2 组合发生器

一些学者研究了这样一些方法，将两个或多个单独的发生器以某种方式结合起来生成

最终的随机数。期望这种组合发生器较之组成它的任何单个发生器有更长的周期，更好的统计性能。当然，采用组合发生器的缺点在于，得到每个 U_i 的代价大于只使用单个发生器的代价。

恐怕最早形式的组合发生器是用第二个 LCG 来对第一个 LCG 的输出洗牌(即打乱)，这些发生器是由 MacLaren 和 Marsaglia(1965)开发的，由 Marsaglia 和 Bray(1968)，Grosenbaugh(1969)，以及 Nance 和 Overstreet(1975)进行了扩展。首先，将第一个 LCG 的前 k 个 U_i(最初建议 $k=128$)依次放入向量 $\boldsymbol{V}=(V_1, V_2, \cdots, V_k)$中。然后，用第二个 LCG 生成一个随机整数 I 均匀分布在整数 1，2，…，k 上(参见第 8.4.2 小节)，并将 V_I 返回作为第一个 $U(0, 1)$变数；接下来，第一个 LCG 用其下一个 U_i 代替 $\boldsymbol{V}$ 中的第 I 个位置，而第二个 LCG 从更新后的向量 $\boldsymbol{V}$ 中随机地选取下一个返回的随机数，以此类推。

洗牌是自然直观的，特别是，因为我们期望它打破任何相关性并大大延长周期。实际上，MacLaren 和 Marsaglia 得到过一个洗牌的发生器具有很好的统计性能，尽管两个单个 LCG 的性能相当差。在后续的评价洗牌的文章中，Nance 和 Overstreet(1978)证实，用一个性能差的 LCG 对另一个性能差的 LCG 洗牌可以得到一个好的组合发生器，例如，延长用于短字长计算机的周期，但是，对一个好的 LCG 进行洗牌，则不明显。另外，他们还发现，向量长度 $k=2$ 时与长得多的向量工作情况一样。

关于这种洗牌办法的几个变种已经得到研究；Bays 和 Dirham(1976)以及 Gebhardt(1967)提出用自身而不是用其他发生器来对一个发生器洗牌。Atkinson(1980)也报告过，对 LCG 的输出只使用一个固定的排序(而不是随机洗牌)“…去掉了几乎是最坏的发生器的影响，并且略微减轻哪怕是可怕的发生器可能引起的损害。”

以某种方式对简单随机数发生器的输出进行重新排列，虽然有这些明显优点，但是人们对于这一类洗过牌的发生器的了解并不够多；例如，除非产生所有中间值，否则无法跳到洗过牌的发生器的输出序列的任意一点，而对 LCG 来说则是可能做到的。

L’Ecuyer(1988)对另一种将两个发生器结合起来的方式进行了讨论和评价[参见 L’Ecuyer和 Tezuka(1991)]。简单来说，其思想是用$\{Z_{1i}\}$和$\{Z_{2i}\}$表示具有不同模数的两个不同 LCG 产生的整数序列，对于某整数个 m，令 $Z_i=(Z_{1i}+Z_{2i})$(对 m 取模)，并最后置$U_i=Z_i/m$。显然，这种思想也可以扩展到多于两个发生器时的情况，且有若干优点；周期很长(在 L’Ecuyer 给出的一个例子中，至少为 10^{18})；每个部件发生器的乘子能够是较小的值(从而事实上提高了在任何计算机上的可移植性和可用性)，得到的发生器计算速度相当快，并且这些发生器的统计性能也非常好。L’Ecuyer 和 Côté(1991)介绍了一个实现这类发生器的可移植的软件包。

L’Ecuyer(1996a，1999a)开发与研究出另一类很有价值的发生器，是以一种特殊的方式将 MRG(在第 7.3.1 节中讨论过的)组合起来得到的。一般说来，同时执行 J 个不同的 MRG [在迭代式(7.2)中使用式(7.3)中的函数 $g()$]，以形成序列$\{Z_{1,i}\}$，$\{Z_{2,i}\}$，…，$\{Z_{J,i}\}$。令 m_1 为式(7.2)用于第一个 MRG 的模，并令 δ_1，δ_2，…，δ_J 为指定的常数，我们定义：

$$Y_i=(\delta_1 Z_{1,i}+\delta_2 Z_{2,i}+\cdots+\delta_J Z_{J,i}) \quad (\text{对 } m_1 \text{ 取模})$$

并最后返回 $U_i=Y_i/m_1$，作为仿真中使用的第 i 个随机数。虽然这类发生器的参数需要经过精心的选择，但有可能得到特别长的周期和期望有好的统计性能。L’Ecuyer(1999a)进一步研究了好的参数设置方法，并针对一些建议的情形给出了小的、易于移植的且速度快的 C 代码；这些情况中最简单的一种情形是将 $J=2$ 个 MRG 组合起来，定义为：

$$Z_{1,i}=(1\,403\,580 Z_{1,i-2}-810\,728 Z_{1,i-3})[\bmod(2^{32}-209)]$$

$$Z_{2,i}=(527\,612 Z_{2,i-1}-1\,370\,589 Z_{1,i-3})[\bmod(2^{32}-22\,853)]$$

$$Y_i=(Z_{1,i}-Z_{2,i})[\bmod(2^{32}-209)]$$

$$U_i=\frac{Y_i}{2^{32}-209}$$

其周期约为 2^{191}(约等于 3.1×10^{57})，并且通过 45 维的检验(见 7.4.2 小节)，有非常好的统计性能。注意，对于这个发生器，种子实际上是一个 6 维向量($Z_{1,0}$，$Z_{1,1}$，$Z_{1,2}$，$Z_{2,0}$，$Z_{2,1}$，$Z_{2,2}$)，并且第一个返回的随机数应当索引为 U_3。在附录 7B 中，我们给出了这种特殊的发生器的可移植的 C 代码实现，它支持大量的流，空间相隔非常远。

这样具有大量流和子流的发生器在 Arena[Kelton 等(2010，第 518 页)]、AutoMod[Banks(2004，第 367 页)]，以及 WITNESS[Lanner(2013)]仿真软件包里已经实现。子流是一个流中的随机数不重叠的相邻子序列。如果人们对其仿真模型进行多次重复运行，则这些软件包将在第 2，3，…次重复运行开始时自动地进入每个流的下一个子流开始处。在使用方差减少技术调用公共随机数时(参见第 11.2.3 小节)，这有助于有关的不同的系统配置之间实现随机数的同步。

Wichmann 和 Hill(1982)提出了如下思路来组合三个发生器，也力争长周期、可移植性、速度，以及在小型计算机上的可用性(以及良好的统计特性)，如果 U_{1i}，U_{2i} 和 U_{3i} 为三个独立的发生器产生的第 i 个随机数，则令 U_i 是 $U_{1i}+U_{2i}+U_{3i}$ 的小数部分(即抛弃十进制小数点左侧的所有数字)；其基本动因参见习题 7.13。这确实产生了一个非常长的周期[虽然没有像原始文章中所宣称的那么长，参见 Wichmann 和 Hill(1984)]，并且具有很好的可移植性和效率。但是，McLeod(1985)后来指出他们的代码在某些计算机体系结构中会有数值困难。Zeisel(1986)随后证明这种发生器与乘 LCG 相同，但这种等价的 LCG 有非常大的模数和乘子；因此，Wichmann-Hill 发生器提供了一种即使在最小的计算机上实现具有很大参数的乘 LCG 的方法。Wichmann-Hill 发生器在 Microsoft Excel 2010 得到实现。

将单个随机数发生器结合起来有很多可能的方法，我们在上面回顾了其中一部分。其他的讨论见 Collins(1987)、Fishman(1996，第 634-645 页)、Gentle(2003，第 46-51 页)，以及 L'Ecuyer(1994b)。

7.3.3 反馈移位寄存器发生器

基于 Tausworthe(1965)的文章已经开发出若干型的发生器。这些发生器与密码方法相关，它们直接对位操作以形成随机数。

用以下递推定义一个二进制数序列 b_1，b_2，…：

$$b_i = (c_1 b_{i-1} + c_2 b_{i-2} + \cdots + c_q b_{i-q}) \quad (\text{对 2 取模}) \tag{7.4}$$

其中，c_1，c_2，…，c_{q-1} 为等于 0 或 1 的常数，且 $c_q=1$。

注意上面 b_i 的递推与式(7.3)的相似性。在 Tausworthe 发生器的大多数应用中，为了计算的简单性，系数 c_j 只有两个为非 0，在这种情况下，式(7.4)变为：

$$b_i = (b_{i-r} + b_{i-q}) \quad (\text{对 2 取模}) \tag{7.5}$$

其中，整数 r 和 q 满足 $0<r<q$。

注意到模 2 加等价于按位异或指令，据此式(7.5)的执行得到加速。即式(7.5)可表示为：

$$b_i = \begin{cases} 0, & b_{i-r} = b_{i-q} \\ 1, & b_{i-r} \neq b_{i-q} \end{cases}$$

这记作 $b_i = b_{i-r} \oplus b_{i-q}$。$\{b_i\}$序列 b_1，b_2，…，b_q 的初始化必须以某种方法来指定，这类似于指定 LCG 的种子 Z_0。

为了形成二进制整数序列 W_1，W_2，…，我们将 l 个连续的 b_i 排列在一起并视其为一个以 2 为底的数，即

$$W_1 = b_1 b_2 \cdots b_l$$

以及

$$W_i = b_{(i-1)l+1} b_{(i-1)l+2} \cdots b_{il}, i = 2,3,\cdots$$

注意，W_i 的递推与式(7.5)中 b_i 的递推是相同的，即

$$W_i = W_{i-r} \oplus W_{i-q} \tag{7.6}$$

其中，“异或”运算是按位执行的。那么第 i 个 $U(0,1)$随机数 U_i 定义为：

$$U_i = \frac{W_i}{2^l}, i = 1, 2, \cdots$$

$\{b_i\}$序列的最大周期为 2^q-1，因为 b_{i-1}，b_{i-2}，…，b_{i-q}能够取得 2^q 种不同的可能状态，出现 q 元组 0，0，…。0 会引起 b_i 序列停留在该状态。令

$$f(x) = x^q + c_1 x^{q-1} + \cdots + c_{q-1} x + 1$$

为式(7.4)给出的递推的特征多项式。Tausworthe(1965)证明了当且仅当 $f(x)$为 Galois 域 F_2 上的本原多项式时，$\{b_i\}$的周期事实上是 2^q-1[见 Knuth(1998a，pp. 29-30)]。Galois 域 F_2 是{0，1}集合，在该集合上定义了模 2 加和乘的二进制运算。如果 l 是对 2^q-1 的素数，则 W_i(以及 U_i)的周期也等于 2^q-1。因此，对于实际的数据存储为 31 位的计算机来说，最长周期等于 $2^{31}-1$，它与 LCG 的周期相同。

例 7.3 令式(7.5)中 $r=3$，$q=5$，且令 $b_1=b_2=\cdots=b_5=1$。因此，对于 $i\geqslant 6$，b_i 为 b_{i-3}和 b_{i-5}的“异或”。在这种情形下，$f(x)$为三项式 x^5+x^2+1，事实上，该式为 F_2 上的本原多项式。那么前 40 个 b_i 是：

11111 00011011101010 00010010110011111 0001

注意，这些位的周期为 $31=2^5-1$，因为 b_{32} 到 b_{36} 与 b_1 到 b_5 相同。若 $l=4$(它是相对 31 的素数)，则可以得到如下的 W_i 序列：

15,8,13,13,4,2,5,9,15,1…

周期也为 31(见习题 7.15)。W_i除以 16 等于 24 便可得到相应的 U_i。◀

之所以提出用 b_i 作为 $U(0,1)$随机数的源，是因为发现式(7.4)给出的递推可以在二进制计算机上利用称为线性反馈移位寄存器(linear feedback shift register，LFSR)的开关电路来实现。这是一个 q 个二进制位的数组进行移位，譬如说，每次向左移一位，向左移出的那一位与数组中的其他位结合形成新的最右边一位。因为式(7.4)与反馈移位寄存器之间的相似性，Tausworthe 发生器又称为 LFSR 发生器。

例 7.4 例 7.3 中讨论的发生器能够用图 7.1 所示的 LFSR 表示。使用取“异或”运算将 b_{i-3}和 b_{i-5}位结合得到一个新的二进制位进入数组最右边的位置(即之前包含的 b_{i-1}位)。位于数组最左边的位置的那一位(即 b_{i-5})从数组移走。表 7.3 给出了 $i=6$，7，…，15 时的 b_{i-5}，b_{i-4}，b_{i-3}，b_{i-2}，以及 b_{i-1}的值。

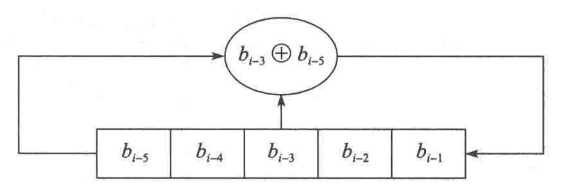


图 7.1　对应递推式(7.5)中 $r=3$ 与 $q=5$ 的 LFSR

表 7.3　$r=3$，$q=5$ 的 LFSR 的连续状态

i	$b_{i-5}^{\dagger}$	b_{i-4}	$b_{i-3}^{\dagger}$	b_{i-2}	b_{i-1}
6	1	1	1	1	1
7	1	1	1	1	0
8	1	1	1	0	0
9	1	1	0	0	0
10	1	0	0	0	1
11	0	0	0	1	1

（续）

i	$b_{i-5}^{\dagger}$	b_{i-4}	$b_{i-3}^{\dagger}$	b_{i-2}	b_{i-1}
12	0	0	1	1	0
13	0	1	1	0	1
14	1	1	0	1	1
15	1	0	1	0	1

† 源位

但是，LFSR 发生器的统计性能差，正如 Matsumoto 和 Kurita(1996)，以及 Tezuka (1995)讨论过的那样。然而，L'Ecuyer(1996b，1999b)考虑了将 LFSR 发生器结合起来，得到的发生器具有较好统计性能和较长的周期。

Lewis 和 Payne(1973)介绍了 LFSR 发生器的一个改进，称为广义反馈移位寄存器(generalized feedback shift register，GFSR)发生器。为了得到 l 位的二进制整数序列 Y_1，Y_2，…，由式(7.5)产生的二进制数序列 b_1，b_2，…，用于填充要生成的整数的第一个(最左边)二进制位。然后，同一个二进制数序列，但滞后 d 位，用于填充该整数的第二个二进制位，也就是说，b_{1+d}，b_{2+d}，…各位用于第二个二进制位。最后，$b_{1+(l-1)d}$，$b_{2+(l-1)d}$，…各位用于填充该整数的第 l 个二进制位。如果 Y_1，Y_2，…，Y_q 为线性独立的，则 Y_i 的周期为 2^q-1，即对于 $a_j=0$ 或 1，有 $a_1Y_1+a_2Y_2+\cdots+a_qY_q=0$，则意味着所有 a_j 等于 0。还要注意 Y_i 满足递推式：

$$Y_i = Y_{i-r} \oplus Y_{i-q} \tag{7.7}$$

其中，“异或”运算按位执行。

下面的例子详细说明 GFSR 发生器是如何工作的。

例 7.5 对于例 7.3 的 LFSR 发生器，$l=4$，且滞后 $d=6$，表 7.4 给出所得到的 GFSR 发生器产生的 Y_i。注意，Y_i 的周期为 31，因为 Y_{32} 到 Y_{36} 与 Y_1 到 Y_5 完全一样。在长度为 31 的周期内，1～15 的每个整数出现两次，而 0 出现一次。

表 7.4　$r=3$，$q=5$，$l=5$ 且 $d=6$ 的 GFSR 发生器

i	Y_i	i	Y_i	i	Y_i
1	1011=11	13	1100=12	25	0010=2
2	1010=10	14	1000=8	26	1001=9
3	1010=10	15	0011=3	27	0101=5
4	1100=12	16	1001=9	28	1101=13
5	1110=14	17	0011=3	29	1110=14
6	0001=1	18	1111=15	30	0111=7
7	0110=6	19	0001=1	31	0100=4
8	0100=4	20	0000=0	32	1011=11
9	1101=13	21	0110=6	33	1010=10
10	1000=8	22	0010=2	34	1010=10
11	0101=5	23	1111=15	35	1100=12
12	1011=11	24	0111=7	36	1110=14

由于 Y_i 的二进制位具有并行属性，不论 l 与 q 的关系如何，GFSR 发生器都可以取 l 等于计算机的字长。如果 $l<q$，则 Y_i 就会多次重复出现，但周期仍为 2^q-1。因此，通过将 q 取非常大的值的办法，就能在 l 位的计算机上得到很长的周期。例如，Fushimi(1990)

研究出了GFSR发生器的一个变种，周期达到 $2^{521}-1$。

Matsumoto 和 Kurita(1992，1994)介绍了旋转的GFSR发生器(twisted GFSR，TGFSR)的思想，其中递推式(7.7)用下式代替：

$$\boldsymbol{Y}_i = \boldsymbol{Y}_{i-r} \oplus \boldsymbol{A}\boldsymbol{Y}_{i-q} \tag{7.8}$$

其中，这里 $\boldsymbol{Y}_i$ 考虑为 $l\times 1$ 维的向量；而 $\boldsymbol{A}$ 是 $l\times l$ 维的矩阵。

两者的元素由0和1组成[如果 $\boldsymbol{A}$ 为单位矩阵，递推式(7.8)与递推式(7.7)相同]。通过合理选择 r，q 以及 $\boldsymbol{A}$，TGFSR发生器的最大周期可达到 $2^{ql}-1$，与之相比，GFSR发生器的周期为 2^q-1(两者都需要 ql 位来存储发生器的状态)。Matsumoto 和 Kurita(1994)讨论了如何选择 $\boldsymbol{A}$ 能够使得TGFSR发生器有好的统计性能。文献号为MT19937的梅森旋转算法(Mersenne twister)[见 Matsumoto 和 Nishimura(1998)]是TGFSR发生器的一种变异，在623维其周期达到了令人惊讶的 $2^{19,937}-1$，并且一般具有很好的统计性能[见第7.4.2小节]。它已经相当流行，并且也已经在Simio语言[Simio(2013)]上实施。叫梅森旋转演算法的原因是 $2^{19,937}-1$ 是梅森素数，即具有 2^p-1 形式的素数(注意，$2^{31}-1$ 也是素数)。

Panneton 等(2006)称如果梅森旋转算法最初阶段碰巧在只有少数设置为1，则在接下来数千个步骤中产生的 U_i 的平均值可能要远小于0.5。随后，他们开发了一类类似的发生器，称为 well equidistributed long period linear(WELL)，其具有大的周期(例如，$2^{19,937}$)、很好的统计性能，以及很少依赖于坏的初始阶段的特性。参见 L'Ecuyer 和 Panneton 的综述论文(2009)。

7.4 随机数发生器的检验

正如我们在7.1节～7.3节已经看到的，目前在计算机仿真中所使用的所有随机数发生器实际上完全是确定的。因此，我们只能希望生成的 U_i 看起来它们像是IID $U(0, 1)$随机变数。在本节中我们讨论一些能检验随机数发生器的检验方法，以确定所产生的 U_i 与真正IID $U(0, 1)$随机变数的近似程度如何(或者能近似到什么程度)。

大多数计算机都有一个“密封的”随机数发生器作为可用软件的一部分。在这种发生器实际用于仿真前，强烈建议用户搞清楚这是哪种发生器，以及它的参数是什么。除非发生器已经由文献认同(与检验)了它是一个“好”发生器(或者是上面推荐的专门的发生器之一)，负责任的分析人员都应当(至少)使用下面讨论的实验检验方法对其进行检验。

有两类相当不同的检验，将分别在7.4.1小节和7.4.2小节中讨论。其中实验检验是通常的统计检验类，且是基于发生器产生的实际 U_i 进行的。理论检验不是统计意义下的检验，实际上完全不产生任何 U_i，而是使用发生器的数字参数全局性地对其进行评价。

7.4.1 实验检验

大概检验一个发生器最直接的方法就是使用这个发生器产生一些 U_i，然后统计检查它们类似IID $U(0, 1)$随机数的紧密程度。我们将讨论四种这样的实验检验方法，一些其他检验方法的讨论参见 Banks et al.(2001，第264-284页)、Fishman(1978，第371-386页)、Knuth(1998a，第41-75页)、L'Ecuyer et al.(2000)，以及 L'Ecuyer 和 Simard(2001，2007)。

设计第一个检验用于检查 U_i 是否在0～1之间均匀分布，因而这个检验是之前(在第6.6.2小节)已经见过的所有参数已知的 χ^2 检验的特殊情形。将区间[0，1]分成 k 个等长的子区间，并产生 U_1，U_2，…，U_n(作为一般规则，这里 k 至少为100)。对于 $j=1$，2，…，k，令 f_j 是落在第 j 个子区间中的 U_i 的个数，且令

$$\chi^2=\frac{k}{n}\sum_{j=1}^{k}\left(f_j-\frac{n}{k}\right)^2$$

那么，当 n 很大时，在 U_i 是 IID $U(0,1)$ 随机变量的原假设下，χ^2 将服从自由度为 $k-1$ 的 χ^2 分布。于是，如果 $\chi^2>\chi^2_{k-1,1-\alpha}$，则在 α 水平上拒绝该原假设，其中，$\chi^2_{k-1,1-\alpha}$ 是自由度为 $k-1$ 的 χ^2 分布上的 $1-\alpha$ 临界点，因为这里遇到的 k 值恐怕很大，可以采用临界点的近似式：

$$\chi^2_{k-1,1-\alpha}\approx(k-1)\left[1-\frac{2}{9(k-1)}+z_{1-\alpha}\sqrt{\frac{2}{9(k-1)}}\right]^3$$

其中，$z_{1-\alpha}$ 为 $N(0,1)$ 分布的上 $1-\alpha$ 临界点。

例 7.6 对 PMMLCG $Z_i=630\ 360\ 016Z_{i-1} \pmod{2^{31}-1}$ 应用均匀性的 χ^2 检验，如附录 7A 中实现的那样，以默认种子使用流 1。取 $k=2^{12}=4\ 096$（从而 U_i 中最显著的 12 位的均匀性将被检验），且令 $n=2^{15}=32\ 768$。得到 $\chi^2=4\ 141.0$；利用上面计算临界点的近似式，$\chi^2_{4\ 095,0.9}\approx 4\ 211.4$。故均匀性的原假设在 $\alpha=0.10$ 水平上不拒绝。从而，只要 χ^2 检验能够确定，由该发生器产生这特定的 32 768 个 U_i 与期望的真正的 IID $U(0,1)$ 随机变量没有明显的不同。◀

第二个实验检验为连续检验，实际上就是把 χ^2 检验推广到高维。如果 U_i 的确是 IID $U(0,1)$ 随机数，则不重叠的 d 元组为：

$$\boldsymbol{U}_1=(U_1,U_2,\cdots,U_d),\quad \boldsymbol{U}_2=(U_{d+1},U_{d+2},\cdots,U_{2d}),\cdots$$

应该是均匀分布在 d 维单位超立方体 $[0,1]^d$ 上的 IID 随机向量。将 $[0,1]$ 分割成 k 个等距的子区间，并产生 $\boldsymbol{U}_1$，$\boldsymbol{U}_2$，…，$\boldsymbol{U}_n$（需要 nd 个 U_i）。用 $f_{j_1j_2\cdots j_d}$ 表示 $\boldsymbol{U}_i$ 的个数，它的第一个成分落在子区间 j_1 中，第二个成分落在子区间 j_2 中，等等（计算 $f_{j_1j_2\cdots j_d}$ 比预想的要容易，参见习题 7.7）。如果令

$$\chi^2(d)=\frac{k^d}{n}\sum_{j_1=1}^{k}\sum_{j_2=1}^{k}\cdots\sum_{j_d=1}^{k}\left(f_{j_1j_2\cdots j_d}-\frac{n}{k^d}\right)^2$$

则 $\chi^2(d)$ 将会近似服从自由度为 k^d-1 的 χ^2 分布[关于连续检验的进一步讨论，参见 L'Ecuyer，Simard 和 Weggenkittl(2002)]。进行 d 维均匀性的检验的做法与上述一维 χ^2 检验完全相同。

例 7.7 对于 $d=2$，我们检验如下原假设：数对 (U_1,U_2)，(U_3,U_4)，…，(U_{2n-1},U_{2n}) 是在单位正方形上均匀分布的 IID 随机向量。我们使用附录 7A 中的发生器，但是从第二个流开始，产生 $n=32\ 768$ 对 U_i。我们取 $k=64$，于是自由度仍然是 $4\ 095=64^2-1$，且水平 $\alpha=0.05$ 的临界值也不变，为 4 245.0。$\chi^2(2)=4\ 016.5$，表明流 2 的前三分之二在二维情况下均匀性是可接受的（回顾一下 2.3 节，该流的长度为 100 000U_i，且我们这里使用了他们中的 $2n=65\ 536$ 个）。对于 $d=3$，我们使用流 3，取 $k=16$（保持自由度数为 $4\ 095=16^3-1$，且水平 $\alpha=0.05$ 的临界值为 4 245.0），并产生 $n=32\ 768$ 个非重叠的 U_i 三元组。以及 $\chi^2(3)=4\ 174.5$，再一次表明在三维情况下的均匀性是可接受的。◀

我们为什么应关注这类高维的均匀性呢？如果单个随机数 U_i 是相关的，那么 d 维向量 $\boldsymbol{U}_i$ 的分布将偏离均匀性；所以，连续检验提供一种对单个 U_i 是独立的假设的间接检查。例如，如果相邻的 U_i 趋向于正相关，则数对 (U_i,U_{i+1}) 将趋于集中在单位正方形西南到东北的对角线附近，从而使得 $\chi^2(2)$ 变大。最后，显然，$d>3$ 的连续检验可能需要大量的内存以计算 $f_{j_1j_2\cdots j_d}$ 的 k^d 个值（在例 7.7 中，$d=3$ 时，取 $k=16$ 对划分 $[0,1]$ 来说恐怕不够细）。

我们考虑的第三个实验检验称为游程（或上升游程）检验，是对独立性假设的更直接的

检验(实际上，它只检验独立性，即特别是我们不检验均匀性)。我们检查 U_i 序列(或者，等价地，检查 Z_i 序列)的办法是把它分成许多个尽可能长的子序列，使得每个子序列内 U_i 的值都是单调增加的，称一个子序列为一个上升游程。例如，考虑如下序列 U_1，U_2，…，U_{10}：0.86，0.11，0.23，0.03，0.13，0.06，0.55，0.64，0.87，0.10。该序列的第一个上升游程的长度为 1(0.86)，接下来的上升游程的长度为 2(0.11，0.23)；然后另一个上升游程的长度为 2(0.03，0.13)；然后上升游程的长度为 4(0.06，0.55，0.64，0.87)；以及最后另一个游程长度为 1(0.10)。对于 n 个 U_i 的序列，计算长度等于 1，2，3，4，5，以及大于或等于 6 的上升游程的个数，并定义

$$r_i = \begin{cases} \text{长度等于 } i \text{ 的上升游程个数}, & i = 1,2,\cdots,5 \\ \text{长度} \geqslant 6 \text{ 的上升游程个数}, & i = 6 \end{cases}$$

(计算 r_i 的算法参见习题 7.8 中，对于上面的 10 个 U_i，$r_1=2$，$r_2=2$，$r_3=0$，$r_4=1$，$r_5=0$ 以及 $r_6=0$)。那么，检验统计为：

$$R = \frac{1}{n}\sum_{i=1}^{6}\sum_{j=1}^{6} a_{ij}(r_i - nb_i)(r_j - nb_j)$$

其中，a_{ij} 为下列矩阵中的第$(i,\ j)$个元素：

$$\begin{bmatrix} 4\,529.4 & 9\,044.9 & 13\,568 & 18\,091 & 22\,615 & 27\,892 \\ 9\,044.9 & 18\,097 & 27\,139 & 36\,187 & 45\,234 & 55\,789 \\ 13\,568 & 27\,139 & 40\,721 & 54\,281 & 67\,852 & 83\,685 \\ 18\,091 & 36\,187 & 54\,281 & 72414 & 90\,470 & 111\,580 \\ 22\,615 & 45\,234 & 67\,852 & 90\,470 & 113\,262 & 139\,476 \\ 27\,892 & 55\,789 & 83\,685 & 111\,580 & 139\,476 & 172\,860 \end{bmatrix}$$

且 b_i 的值如下：

$$(b_1, b_2, \cdots, b_6) = \left(\frac{1}{6}, \frac{5}{24}, \frac{11}{120}, \frac{19}{720}, \frac{29}{5040}, \frac{1}{840}\right)$$

[关于这些常数[⊖] * 的推导过程，参见 Knuth(1998a，第 66-69 页)。上面给出的 a_{ij} 为精确到 5 位有效数字]。对于大的 n(Knuth 建议取 $n \geqslant 4\,000$)，在 U_i 是 IID 随机变量的原假设下，R 将近似服从自由度为 6 的 χ^2 分布。

例 7.8 对附录 7A 中的发生器的流 4 进行上升游程检验，采用 $n=5\,000$，且得到$(r_1, r_2, \cdots, r_6)=(808, 1026, 448, 139, 43, 4)$，得到 $R=9.3$。因为 $\chi^2_{6,0.90}=12.6$。我们在 $\alpha=0.05$ 水平上不拒绝独立性的假设。◀

上升游程检验可以以显而易见的方式反过来得到下降游程检验，a_{ij} 与 b_i 常数相同。还有一些其他类型的游程检验，例如在同一个序列中统计上升游程或下降游程，或者不考虑游程的长度而只统计游程个数。建议读者参考文献，例如，Banks 等(2001，第 270-278 页)以及 Fishman(1978，第 373-376 页)。回顾 6.3 节中讨论的游程检验。因为游程检验只能检查独立性(不特别地检查均匀性)，在进行 χ^2 检验或者连续检验之前进行游程检验恐怕是一个好主意，原因是 χ^2 检验和连续检验都隐含独立性的假设。

考虑的最后一种实验检验是一种直接的方法以评价所产生的 U_i 是否存在可辨别的相关性：对于某个 l，只计算其滞后 $j=1, 2, \cdots, l$ 的相关系数的估计。回忆一下，由第 4.3 节，随机变量的序列 X_1，X_2，…的滞后 j 的相关系数定义为 $\rho_j=C_j/C_0$，其中，

$$C_j = \text{cov}(X_i, X_{i+j}) = E(X_i, X_{i+j}) - E(X_i)E(X_{i+j})$$

它是序列中相隔 j 的项之间的协方差系数；注意，$C_0=\text{var}(X_i)$(这里假设该过程是协方差

⊖ * Knuth D. E.，计算机编程艺术，第 2 卷第 67 页，

平稳的；参见第 4.3 节）。在现在的情况下，我们关心的是 $X_i=U_i$，并且假设所有 U_i 均匀分布在[0, 1]区间上，有 $E(U_i)=\frac{1}{2}$，且 $\text{var}(U_i)=\frac{1}{12}$，从而 $C_j=E(U_iU_{i+j})-\frac{1}{4}$，且 $C_0=\frac{1}{12}$。因此，在这种情况下，$\rho_j=12E(U_iU_{i+j})-3$。由所产生的值的序列 U_1，U_2，…，U_n，通过直接由 U_1，U_{1+j}，U_{1+2j} 等等来估计 $E(U_iU_{i+j})$，就可得到 ρ_j 的估计值为：

$$\hat{\rho}_j=\frac{12}{h+1}\sum_{k=0}^{h}U_{1+kj}U_{1+(k+1)j}-3$$

其中，$h=\lfloor(n-1)/j\rfloor-1$。

在进一步假设 U_i 独立的条件下，则得到[例子参见 Banks 等(2001，第 279 页)]：

$$\text{var}(\hat{\rho}_j)=\frac{13h+7}{(h+1)^2}$$

在原假设 $\rho_j=0$ 下，并假设 n 很大，可以证明检验统计量

$$A_j=\frac{\hat{\rho}_j}{\sqrt{\text{var}(\hat{\rho}_j)}}$$

近似服从标准正态分布。这提供了一种在 α 水平上滞后 j 相关系数为 0 的检验，如果 $|A_j|>z_{1-\alpha/2}$，则拒绝该假设。这个检验恐怕应当针对若干个 j 值进行，因为可能，例如，在滞后 1 或 2 时不存在显著的相关性，由于发生器的某些异常，而在滞后 3 时 U_i 之间存在依赖性。

例 7.9 分别检验过附录 7A 中的发生器的第 5～10 个流的滞后 1～6 的相关性，每种情形取 $n=5\,000$，即我们检验第 5 个流滞后 1 的相关系数，第 6 个流滞后 2 的相关系数，等等。A_1，A_2，…，A_6 的值分别为 0.90，−1.03，−0.12，−1.32，0.39 和 0.76，与 $N(0, 1)$分布比较，在 $\alpha=0.05$(或者更小)水平上，没有一个显著偏离于 0。因此，这些流的前 5 000 个随机数没有出现这些滞后的显著的自相关性。◀

正如前面提到的，这些仅仅是许多可能的实验检验中的四个。例如，可以用第 6.6.2 小节中介绍的科尔莫戈罗夫-斯米尔洛夫检验(对于所有参数已知的情况)来代替检验一维均匀性的 χ^2 检验。很多实验检验方法都是围绕下述思想开发出来的，即待检验的发生器用于仿真一个已知(总体)性能指标的相对简单的随机系统，仿真估计这些性能指标。将仿真结果以某种方式与已知的准确的“答案”进行比较，可能通过 χ^2 检验方式。习题 7.10 就是这种思想的一个简单应用；Rudolph 和 Hawkins(1976)通过使用一些发生器来仿真马尔可夫(Markov)过程来检验它们。通常而言，我们觉得执行实验检验的次数应该切实可行。在这一点上，应该指出的是有若干综合性检验包来评价随机数发生器，包括：DIEHAED[Marsaglia(1995)]、NIST 检验包[Rukhin 等(2001)]，以及 Test$U$01 [L'Ecuyer 和 Simard(2005)]，最后一个检验包中提供了超过 60 种实验检验方法。Gentle (2003，第 79-85 页)也介绍了这些检验包。

为避免读者留下这样一个印象，我们本节提供的实验检验方法完全缺乏判断力，我们对著名的随机数发生器 RANDU[定义为 $Z_i=65\,539Z_{i-1}(\text{mod }2^{31})$]，种子为 $Z_0=123\,456\,789$，进行了如在例 7.6 至例 7.9 中一样检验。检验统计量如下。

χ^2 检验：　　$\chi^2=4\,202.0$

连续检验：　　$\chi^2(2)=4\,202.3$

　　　　　　　$\chi^2(3)=16\,252.3$

上升游程检验：$R=6.3$

相关性检验：　所有阶下的 A_j 都不显著

虽然均匀性在区间[0, 1]和单位正方形中是可接受的，注意，三维连续检验统计量的

值很大，表明根据单位立方体下的均匀性，该发生器有严重问题。RANDU 是一个有致命缺陷发生器，主要是由于在三维下它完全失败了，其中的原因将会在 7.4.2 小节中讲到。

实验检验的一个潜在缺陷是，它们总是局部的；也就是说，仅仅对循环中的一段(例如，对 LCG 来说)进行了检验，这一段实际上是用于产生检验用的 U_i。这样，关于该发生器在其循环的其他段的性能会怎么样无法说出什么来。从另一方面，实验检验的这种局部性质可能是一个优点，因为它使我们可以检查以后在仿真中使用的实际随机数(往往通过分析模型的运行和用于产生必须随机数的技术，我们事先能够计算出仿真时会用多少随机数，或者至少能得到一个保守的估计值)。然后，这整个随机数流能实验检验，人们希望无额外的费用(例 7.6 至例 7.9 中的检验都在一个程序中全部完成，并且在一个旧的、性能一般的计算机上所花费的时间仅需几秒钟而已)。通过多次重复整个检验，并将检验统计的观测值与原假设下的分布进行统计比较，就能进行更全局的实验检验，Fishman(1978，第 371-372 页)建议这种方法。比如，例 7.8 的上升游程检验可以，做 100 次，使用同一个发生器的 100 个独立的随机数流，每个长度为 5 000。这会得到 100 个独立 R 值，然后，使用，例如，所有参数已知的 K-S 检验，将它们与 6 自由度的 χ^2 分布进行比较。

7.4.2　理论检验

现在讨论随机数发生器的理论检验。因为这些检验相当复杂以及数学上的复杂性，将稍微定性地讨论；关于详细的内容，请参见文献 Fishman(2006，第 119-123 页，第 133-134 页，第 140-141 页)、Knuth(1998a，第 80-115 页)以及 L'Ecuyer(1998，第 106-114 页)。正如前面所提到的，理论检验不需要产生任何 U_i，而是先验估计，即通过观察发生器的结构并定义常数，这些检验告知一个发生器的性能好坏。理论检验与实验检验还有一个区别在于它是全局性的，即检查在发生器整个循环上的行为特性。如同在 7.4.1 小节结束时所提到的，究竟局部检验和全局检验哪一个更好是可以讨论的；全局检验有自然的吸引力，但一般并不告知一个循环的特定段的行为特性如何。

有时有可能直接由定义发生器的常数来计算在整个循环上的“样本”均值、方差和相关系数。Kennedy 和 Gentle(1980，第 139-143 页)给出了很多这方面的结果。例如，在一个满周期 LCG 中，在整个循环上所取的 U_i 的均值为 $\frac{1}{2}-\frac{1}{2m}$，显然，典型的情况，如果 m 取值很大(数十亿)，这个均值非常接近所希望的 $\frac{1}{2}$；参见习题 7.14。类似地，能计算在整个满循环上的 U_i 的“样本”方差等于 $\frac{1}{12}-\frac{1}{12m^2}$，它接近 $U(0,1)$ 分布的方差 $\frac{1}{12}$。Kennedy 和 Gentle 也讨论了 LCG 的“样本”相关系数。虽然所得到的公式似乎令人满意，但他们有可能是误导；例如，对满周期 LCG 样本滞后 1 相关系数的结果最小化，建议 a 的选择是接近 $\sqrt{m}$，从其他重要的统计考虑的角度来看，得到的是一个差的选择。

最著名的理论检验是基于 Marsaglia(1968)的观测——“大部分随机数都落在平面上”。也就是说，如果 U_1，U_2，…是由 LCG 产生的随机数序列，则相互重叠的 d 元组 $(U_1, U_2, \cdots, U_d)$，$(U_2, U_3, \cdots, U_{d+1})$，…将会落在少数几个穿越 d 维单位超立方体 $[0,1]^d$ 的 $(d-1)$ 维超平面上。例如，若 $d=2$，则随机数对 (U_1, U_2)，(U_2, U_3)，…的排列将以“格子”方式沿着若干不同族的平行线穿过单位正方形。同族的线是平行的，但是不同族的线是不平行的。

在图 7.2 和图 7.3 中，我们分别显示了满周期乘 LCG $Z_i=18Z_{i-1}(\text{mod } 101)$ 和 $Z_i=$

$2Z_{i-1}$(mod 101)所有可能的(U_i，U_{i+1})对(因为模数 $m=101$ 为素数，并且每个乘子 a 都为模 m 的素元，所以每种情形周期都等于 100)。图 7.2 所示的有明显的规则性，虽然这看上去肯定不是非常“随机”，但扰动不是太严重，因为这些数对相当好地填满了单位正方形，或者，对于如此小的模数，至少说能够期望的也就这样。另一方面，图 7.3 中的全部 100 个数对仅仅落在了两条平行线上，这肯定是不能令人满意的。因为在单位正方形中存在大片区域，这些区域我们未能实现 U_i 对，使用这样的发生器进行仿真，几乎肯定会产生无效的结果。

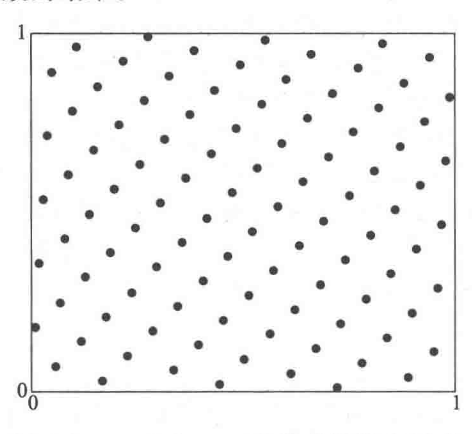


图 7.2　$m=101$，$a=18$ 的满周期 LCG 的二维格结构

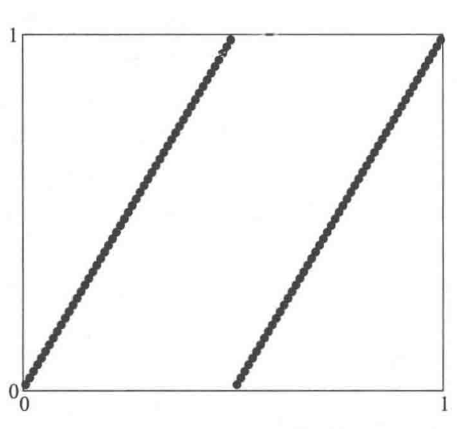


图 7.3　$m=101$，$a=2$ 的满周期 LCG 的二维格结构

三维存在同样的困难。图 7.4 显示了用乘 LCG 发生器 RANDU($m=2^{31}$，$a=65\,539$)产生的 2 000 个三元组(U_i，U_{i+1}，U_{i+2})，从立方体外某个特定点观察。注意，U_i的所有三元组仅仅落在穿越单位立方体的 15 个平行平面上(一般说来，整个周期内近五亿个三元组落在这些相同的平行平面上)。这也解释了在第 7.4.1 小节的末尾所说的 RANDU 三维连续检验性能很差的原因。

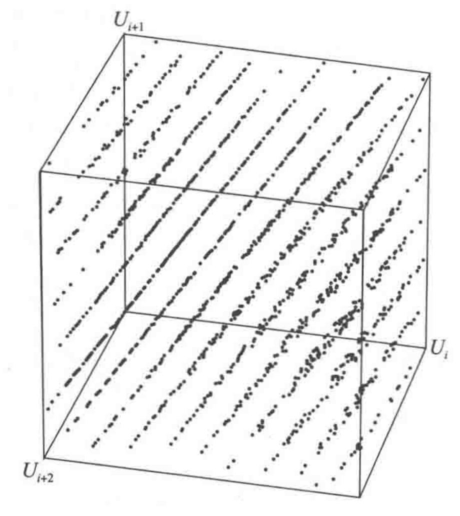


图 7.4　$m=2^{31}$，$a=65\,539$ 的乘 LCG RANDU 的 2 000 个三元组的三维格结构

在那些覆盖所有重叠 d 元(U_i，U_{i+1}，…，U_{i+d-1})的平行超平面所有族中，找出相邻超平面距离最大的，并将该距离记为 $\delta_d(m, a)$。这个计算 $\delta_d(m, a)$的思想是由 Coveyou 和 MacPherson(1967)提出的，被典型地称为谱检验。如果 $\delta_d(m, a)$小，则我们可以期望相应的随机数发生器能够均匀地填满 d 维单位超立方体$[0, 1]^d$。

对于 LCG，有可能计算出 $\delta_d(m, a)$的理论下界 $\delta_d^*(m)$，由下式给出：

$$\delta_d(m,a) \geqslant \delta_d^*(m) = \frac{1}{\gamma_d m^{1/d}} \quad (\text{对于所有 } a)$$

其中，γ_d 为常数，只有当 $d \leqslant 8$ 时其准确值已知。

然后我们可以定义 LCG 的如下性能指数：

$$S_d(m,a) = \frac{\delta_d^*(m)}{\delta_d(m,a)}$$

以及

$$M_8(m,a) = \min_{2 \leqslant d \leqslant 8} S_d(m,a)$$

这些性能指数将介于 0～1 之间，希望接近 1。文献 Knuth(1998a，第 98-104 页)以及 L'Ecuyer和 Couture(1997)讨论了进行谱检验的方法(即计算 $\delta_d(m, a)$)。

表 7.5 给出了一些众所周知的 LCG 的 $S_d(m, a)$和 $M_8(m, a)$值，表中第 2 列和第 3 列乘子分别为 a_1 和 a_2 的 LCG 在 7.2.2 小节中讨论过，且看到，对于谱检验来说，第二个发生器性能更好。第 4 列中的 a=742 938 285 的发生器是由 Fishman 和 Moore(1986)通过大量的比较找到的，对准则 $M_6(2^{31}-1, a)$来说，该发生器是最好的。最后，第 5 列是 RANDU 的谱检验结果，其值 $S_3(2^{31}, 65\ 539)=0.012$ 清楚地反映了其在 $d=3$ 维时表现不佳。

表 7.5　一些 LCG 的谱检验结果

性能指数	发生器			
	$m=2^{31}-1$ $a=a_1=16\ 807$	$m=2^{31}-1$ $a=a_2=630\ 360\ 016$	$m=2^{31}-1$ $a=742\ 938\ 285$	$m=2^{31}$ $a=65\ 539$
$S_2(m, a)$	0.338	0.821	0.867	0.931
$S_3(m, a)$	0.441	0.432	0.861	0.012
$S_4(m, a)$	0.575	0.783	0.863	0.060
$S_5(m, a)$	0.736	0.802	0.832	0.157
$S_6(m, a)$	0.645	0.570	0.834	0.293
$S_7(m, a)$	0.571	0.676	0.624	0.453
$S_8(m, a)$	0.610	0.721	0.707	0.617
$M_8(m, a)$	0.338	0.432	0.624	0.012

7.4.3　关于检验的某些一般看法

从数量、种类，以及复杂程度而言，随机数发生器的检验的确令人困惑。而且更糟糕的是，一直有(并且恐怕永远有)大量的争论究竟哪些检验方法最好，理论检验是否真的比实验检验更权威，诸如此类。诚然，即使再多的检验也不能让每一个人坚信某个特定的发生器绝对是“最好的”。然而，一个通常的建议是，随机数发生器的检验方法要与它的用途相一致。例如，假如在仿真中习惯成对使用随机数，则需要检验 U_i对的性能(恐怕用二维连续检验)。在更广的意义下，这条建议意味着，如果使用随机数的仿真成本很高，需要高精度的结果，或者仿真是一个较大研究的特别关键的部分，人们应该更认真地选择和检验随机数发生器。

附录 7A　PMMLCG 的可移植 C 源码

这里给出 PMMLCG 的 C 语言计算机代码，该发生器的定义是：模数 $m=m*=2^{31}-1=2\ 147\ 483\ 647$，乘子 $a=a_2=630\ 360\ 016$，在 7.2.2 小节末尾讨论过。这里给出的代码可以从网站 www.mhhe.com/law 上下载。这些代码紧密地基于 Marse 和 Roberts(1983)给出的 FORTRAN 代码，并且它要求正确地表示与计算 $-m^*$ 到 m^* 之间的整数。该发生器具有 100 个不同的数流，相隔 100 000 个。

通常不建议将这个发生器用于严格的实际问题，因为附录 7B 中的组合 MRG 具有更好的统计性能。

图 7.5 给出了该随机数发生器的 ANSI-标准 C 语言(即使用函数原型)版本的代码，包括三个函数，注释中已详细说明。图 7.6 给出了头文件(lcgrand.h)，用户必须使用 #include来声明这些函数。我们已经在多种计算机和编译器上使用过这些代码，而且它已经用于第 1、2 章的 C 语言例子中。

```
/* Prime modulus multiplicative linear congruential generator
   Z[i] = (630360016 * Z[i-1]) (mod(pow(2,31) - 1)), based on Marse and Roberts'
   portable FORTRAN random-number generator UNIRAN.  Multiple (100) streams are
   supported, with seeds spaced 100,000 apart.  Throughout, input argument
   "stream" must be an int giving the desired stream number.  The header file
   lcgrand.h must be included in the calling program (#include "lcgrand.h")
   before using these functions.

   Usage: (Three functions)

   1. To obtain the next U(0,1) random number from stream "stream," execute
          u = lcgrand(stream);
      where lcgrand is a float function.   The float variable u will contain the
      next random number.

   2. To set the seed for stream "stream" to a desired value zset, execute
          lcgrandst(zset, stream);
      where lcgrandst is a void function and zset must be a long set to the
      desired seed, a number between 1 and 2147483646 (inclusive).  Default
      seeds for all 100 streams are given in the code.

   3. To get the current (most recently used) integer in the sequence being
      generated for stream "stream" into the long variable zget, execute
          zget = lcgrandgt(stream);
      where lcgrandgt is a long function. */

/* Define the constants. */

#define MODLUS 2147483647
#define MULT1       24112
#define MULT2       26143

/* Set the default seeds for all 100 streams. */

static long zrng[] =
{         1,
 1973272912, 281629770,  20006270,1280689831,2096730329,1933576050,
  913566091, 246780520,1363774876, 604901985,1511192140,1259851944,
  824064364, 150493284, 242708531,  75253171,1964472944,1202299975,
  233217322,1911216000, 726370533, 403498145, 993232223,1103205531,
  762430696,1922803170,1385516923,  76271663, 413682397, 726466604,
  336157058,1432650381,1120463904, 595778810, 877722890,1046574445,
   68911991,2088367019, 748545416, 622401386,2122378830, 640690903,
 1774806513,2132545692,2079249579,  78130110, 852776735,1187867272,
 1351423507,1645973084,1997049139, 922510944,2045512870, 898585771,
  243649545,1004818771, 773686062, 403188473, 372279877,1901633463,
  498067494,2087759558, 493157915, 597104727,1530940798,1814496276,
  536444882,1663153658, 855503735,  67784357,1432404475, 619691088,
  119025595, 880802310, 176192644,1116780070, 277854671,1366580350,
 1142483975,2026948561,1053920743, 786262391,1792203830,1494667770,
 1923011392,1433700034,1244184613,1147297105, 539712780,1545929719,
  190641742,1645390429, 264907697, 620389253,1502074852, 927711160,
  364849192,2049576050, 638580085, 547070247 };
/* Generate the next random number. */

float lcgrand(int stream)
{
    long zi, lowprd, hi31;

    zi     = zrng[stream];
    lowprd = (zi & 65535) * MULT1;
    hi31   = (zi >> 16) * MULT1 + (lowprd >> 16);
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    lowprd = (zi & 65535) * MULT2;
    hi31   = (zi >> 16) * MULT2 + (lowprd >> 16);
```

图 7.5　$m=2^{31}-1$，$a=630,360,016$ 的 PMMLCG 的 C 代码，基于 Marse 和 Roberts(1983)

```
    zi     = ((lowprd & 65535) - MODLUS) +
             ((hi31 & 32767) << 16) + (hi31 >> 15);
    if (zi < 0) zi += MODLUS;
    zrng[stream] = zi;
    return (zi >> 7 | 1) / 16777216.0;
}

void lcgrandst (long zset, int stream) /* Set the current zrng for stream
                                          "stream" to zset. */
{
    zrng[stream] = zset;
}

long lcgrandgt (int stream) /* Return the current zrng for stream "stream". */
{
    return zrng[stream];
}
```

图 7.5　(续)

```
/* The following 3 declarations are for use of the random-number generator
   lcgrand and the associated functions lcgrandst and lcgrandgt for seed
   management.  This file (named lcgrand.h) should be included in any program
   using these functions by executing
       #include "lcgrand.h"
   before referencing the functions. */

float lcgrand(int stream);
void  lcgrandst(long zset, int stream);
long  lcgrandgt(int stream);
```

图 7.6　伴随图 7.5 中 C 代码的头文件(lcgrand. h)

附录 7B　组合 MRG 的可移植 C 源码

这里给出一个 ANSI-标准 C 语言函数 mrand()，它实现在 7. 3. 2 小节中说明过的组合 MRG，取自 L'Ecuyer(1999a)，该发生器支持多流(多达 10 000)，种子向量相隔 10^{16}(千万亿)。这些代码要求 $-2^{53}\sim2^{53}$ 之间的所有整数以浮点数的形式精确地表示，字长为 32 位的计算机和符合 IEEE 浮点数存储和运算标准的编译器这种(通常)环境都满足这个要求。这里给出的所有代码都能从网站 www. mhhe. com/law 上下载。

图 7. 7 给出了该发生器的代码，包括三个函数，代码的注释中进行了介绍。图 7. 8 给出了头文件(mrand. h)，用户必须在主函数前以调用函数的方式使用 # include。图 7. 9 给出了文件 mrand-seeds. h 的前 23 行和最后 4 行(说明随机数流 1～20 和 9 998～10 000 的种子向量)。mrand-seeds. h 包含相隔 10^{16} 的 10 000 个流的种子向量。我们已经在多种计算机和编译器上成功地使用过这些代码，虽然某些编译器关于 mrand. h 中数字大小会发出温和的警告信息(通过编译器开关，如某些 UNIX C 编译器中的-w，这种警告通常可以消除)。另外，调用程序时可能必须加载数学程序库(例如，UNIX C 编译器上的-lm 开关)。

```
/* Combined MRG from Sec. 7.3.2, from L'Ecuyer (1999a).  Multiple
   (10,000) streams are supported, with seed vectors spaced
   10,000,000,000,000,000 apart.  Throughout, input argument "stream"
   must be an int giving the desired stream number.  The header file
   mrand_seeds.h is included here, so must be available in the
   appropriate directory.  The header file mrand.h must be included in
   the calling program (#include "mrand.h") before using these
   functions.
```

图 7. 7　伴随图 7. 5 中 C 代码的头文件(lcgrand. h)

```
   Usage: (Three functions)

   1. To obtain the next U(0,1) random number from stream "stream,"
      execute
          u = mrand(stream);
      where mrand is a double function.  The double variable u will
      contain the next random number.

   2. To set the seed vector for stream "stream" to a desired 6-vector,
      execute
          mrandst(zset, stream);
      where mrandst is a void function and zset must be a double
      vector with positions 0 through 5 set to the desired
      stream 6-vector, as described in Sec. 7.3.2.

   3. To get the current (most recently used) 6-vector of integers in
      the sequences (to use, e.g., as the seed for a subsequent
      independent replication), into positions 0 through 5 of the
      double vector zget, execute
          mrandgt(zget, stream);
      where mrandgt is void function.  */

#include "mrand_seeds.h"
#define norm   2.328306549295728e-10  /* 1.0/(m1+1) */
#define norm2  2.328318825240738e-10  /* 1.0/(m2+1) */
#define m1     4294967087.0
#define m2     4294944443.0

/* Generate the next random number. */
double mrand(int stream)
{
    long k;
    double p,
           s10 = drng[stream][0], s11 = drng[stream][1], s12 = drng[stream][2],
           s20 = drng[stream][3], s21 = drng[stream][4], s22 = drng[stream][5];

    p = 1403580.0 * s11 - 810728.0 * s10;
    k = p / m1;  p -= k*m1;  if (p < 0.0) p += m1;
    s10 = s11;   s11 = s12;  s12 = p;

    p = 527612.0 * s22 - 1370589.0 * s20;
    k = p / m2;  p -= k*m2;  if (p < 0.0) p += m2;
    s20 = s21;   s21 = s22;  s22 = p;

    drng[stream][0] = s10;  drng[stream][1] = s11;  drng[stream][2] = s12;
    drng[stream][3] = s20;  drng[stream][4] = s21;  drng[stream][5] = s22;

    if (s12 <= s22) return ((s12 - s22 + m1) * norm);
    else return ((s12 - s22) * norm);
}

/* Set seed vector for stream "stream". */

void mrandst(double* seed, int stream)
{
int i;
    for (i = 0; i <= 5; ++i) drng[stream][i] = seed[i];
}
/* Get seed vector for stream "stream". */

void mrandgt(double* seed, int stream)
{
int i;
    for (i = 0; i <= 5; ++i) seed[i] = drng[stream][i];
}
```

图 7.7 (续)

```
/* Header file "mrand.h" to be included by programs using mrand.c */

double mrand(int stream);
void mrandst(double* seed, int stream);
void mrandgt(double* seed, int stream);
```

图 7.8 伴随图 7.7 中的 C 代码的 C 头文件(mrand.h)

```
/* Header file "mrand_seeds.h" included by mrand.c */

static double drng[][6] =
{
            0,           0,           1,           0,           0,           1,
   1772212344,  1374954571,  2377447708,   540628578,  1843308759,   549575061,
   2602294560,  1764491502,  3872775590,  4089362440,  2683806282,   437563332,
    376810349,  1545165407,  3443838735,  3650079346,  1898051052,  2606578666,
   1847817841,  3038743716,  2014183350,  2883836363,  3242147124,  1955620878,
   1075987441,  3468627582,  2694529948,   368150488,  2026479331,  2067041056,
    134547324,  4246812979,  1700384422,  2358888058,    83616724,  3045736624,
   2816844169,   885735878,  1824365395,  2629582008,  3405363962,  1835381773,
    675808621,   434584068,  4021752986,  3831444678,  4193349505,  2833414845,
   2876117643,  1466108979,   163986545,  1530526354,    68578399,  1111539974,
    411040508,   544377427,  2887694751,   702892456,   758163486,  2462939166,
   3631741414,  3388407961,  1205439229,   581001230,  3728119407,    94602786,
   4267066799,  3221182590,  2432930550,   813784585,  1980232156,  2376040999,
   1601564418,  2988901653,  4114588926,  2447029331,  4071707675,  3696447685,
   3878417653,  2549122180,  1351098226,  3888036970,  1344540382,  2430069028,
    197118588,  1885407936,   576504243,   439732583,   103559440,  3361573194,
   4024454184,  2530169746,  2135879297,  2516366026,   260078159,  2905856966,
   2331743881,  2059737664,   186644977,   401315249,    72328980,  1082588425,
    694808921,  2851138195,  1756125381,  1738505503,  2662188364,  3598740668,
   2834735415,  2017577369,  3257393066,  3823680297,  2315410613,   637316697,
   4132025555,  3700940887,   838767760,  2818574268,  1375004287,  2172829019,
.
.
.
    560024289,  1830276631,   144885590,  1556615741,  1597610225,  1856413969,
   1031792556,  1844191084,  1441357589,  3147919604,   199001354,  2555043119,
   2023049680,  4184669824,  4074523931,   252765086,  3328098427,  1480103038
};
```

图 7.9 伴随图 7.7 和图 7.8 中的 C 代码的种子向量文件(mrand_seeds.h)的摘录(完整的文件可从 www.mhhe.com/law 下载)

该发生器的 C、C++ 和 Java 代码可从 www.iro.umontreal.ca/~lecuyer 下载，该代码提供子流功能(参见 7.3.2 小节，也可参见 L'Ecuyer，Simard，Chen 和 Kelton (2002))。

习题

7.1 仅仅使用纸和笔求例 7.2 的 LCG 的 Z_{500}。

7.2 对下面的乘 LCG，计算足够多的 Z_i 值($i \geqslant 1$)以覆盖整个循环：

(a) $Z_i=(11Z_{i-1}) \pmod{16}$，$Z_0=1$

(b) $Z_i=(11Z_{i-1}) \pmod{16}$，$Z_0=2$

(c) $Z_i=(2Z_{i-1}) \pmod{13}$，$Z_0=1$

(d) $Z_i=(3Z_{i-1}) \pmod{13}$，$Z_0=1$

注意，(a)和(b)的 m 形式为 2^b；(c)为 PMMLCG，因为 $a=2$ 是模数 $m=13$ 的素元。

7.3 不实际计算任何 Z_i，判断下列的混合 LCG 中哪些具有满周期：

(a) $Z_i=(13Z_{i-1}+13) \pmod{16}$

(b) $Z_i=(12Z_{i-1}+13) \pmod{16}$

(c) $Z_i=(13Z_{i-1}+12) \pmod{16}$

(d) $Z_i=(Z_{i-1}+12)\ (\text{mod } 13)$

7.4 对于习题 7.3 中的四个混合 LCG，计算足够多的 Z_i 值($i\geqslant1$)以覆盖一个完整的循环，每种情形均令 $Z_0=1$。请说明结果。

7.5 (a) 使用附录 7A 中的 Marse-Roberts 代码，在你自己的计算机上实现 PMMLCG $Z_i=630\ 360\ 016Z_{i-1}(\text{mod } 2^{31}-1)$。

(b) 对于该发生器，重复例 7.6 至例 7.9 中的实验检验，将你的结果与例中给出的结果进行比较。

7.6 用习题 7.2(a)的乘 LCG 来对习题 7.3(d)的混合 LCG 洗牌，使用长度为 2 的向量 $\mathbf{V}$。令两个 LCG 的种子都是 1，并列出 $\mathbf{V}$，$\boldsymbol{I}$ 和 V_I 的前 100 个值。找出其周期并对结果进行一般性说明。

7.7 对于关于均匀性的 χ^2 检验，校验下面计算 f_1，f_2，…，f_k 的算法：

```
Set f_j = 0 for j = 1,2, . . . , k
For i = 1, . . . , n do
   Generate U_i
   Set J = ⌈kU_i⌉
   Replace f_J by f_J + 1
End do
```

对于实数 x，$\lceil x\rceil$表示大于或等于 x 的最小整数。将此算法推广到一般 d 维连续检验的检验统计量的计算。

7.8 说明下面的算法能够由所产生的数 U_1，U_2，…，U_n 正确计算上升游程检验中的 r_1，r_2，…，r_6：

```
Set r_j = 0 for j = 1, . . . , 6
Generate U_1, set A = U_1, and set J = 1
For i = 2, . . . , n do
   Generate U_i and set B = U_i
   If A ≥ B then
      Set J = min(J,6)
      Replace r_J by r_J + 1
      Set J = 1
   Else
      Replace J by J + 1
   End if
   Replace A by B
End do
Set J = min(J,6)
Replace r_J by r_J + 1
```

7.9 对你的计算机自带的随机数发生器进行 χ^2 检验、二维和三维连续检验、上升游程检验以及滞后 1，2，…，5 的相关性检验。使用例 7.6 到例 7.9 中相同的 n，k 和 α 值。(如果你的发生器通不过这些检验，那么我们建议你使用它时要谨慎，或者查阅文献或者你自己进行进一步检验，直到你能得到关于它的更多信息为止)

7.10 实验性地检验随机数发生器的一般方法是用待检验的发生器来仿真一个参数已知的简单的随机模型，然后使用标准检验将估计值与已知参数进行比对。例如，我们知道，在独立地掷两个公平骰子中，两个骰子点数之和为 2，3，…，12 的概率依次是$\frac{1}{36}$，$\frac{1}{18}$，$\frac{1}{12}$，$\frac{1}{9}$，$\frac{5}{36}$，$\frac{1}{6}$，$\frac{5}{36}$，$\frac{1}{9}$，$\frac{1}{12}$，$\frac{1}{18}$，$\frac{1}{36}$。仿真 1 000 次独立掷一对独立的公平骰子，并用第 6 章中的一个合适的检验方法，将所得点数之和取 2，3，…，12 的次数的观测比例数与已知概率进行比较。请使用你计算机中自带的发生器或附录 7A、7B 中的一个发生器。

7.11 对于习题 7.3(d)中的 LCG，绘制数对(U_1, U_2)，(U_2, U_3)，…的图，并观察所得到的格结构。注意，该 LCG 有满周期。

7.12 考虑第 7.3.1 小节中讨论的 Fibonacci 发生器

(a) 说明该发生器不可能产生三个连续输出值出现如下关系：$U_{i-2}<U_i<U_{i-1}$。

(b) 说明对于一个“完美”的随机数发生器，出现(a)中的关系的概率为 1/6。

这揭示了该发生器的严重缺点，正如 Bratley、Fox 和 Schrage 指出的那样，他们因此获得了 U. Dieter 荣誉。

7.13 假设 U_1，U_2，…，U_k 是 IID $U(0, 1)$随机变量。请证明 $U_1+U_2+\cdots+U_k$ 所得结果的小数部分

(即抛弃十进制小数点左侧的所有数字)在[0, 1]区间上也是均匀分布的[这一性质推动了Wichmann和Hill(1982)的组合随机数发生器的出现]。

7.14 请证明，对于一个满周期LCG，在一个整循环内取得的U_i的均值为$\frac{1}{2}-\frac{1}{2m}$。[提示：对于一个正整数k，根据欧拉公式有$1+2+\cdots+k=k(k+1)/2$]

7.15 对于例7.3，利用式(7.6)证明W_i的周期为31。

第8章 随机变量的产生

8.1 引言

具有任何随机现象的仿真从根本上说一定包含着从概率分布中采样或产生随机变量。如在第7章那样，我们使用短语“产生一个随机变量”来表述从期望分布中获得一个随机变量的观察结果(或一个实现)的活动。这些分布往往定义为对观测数据拟合某种合适分布形式的结果，例如指数分布、伽马分布、泊松分布等，如第6章所讨论的那样。本章我们假定已经用某种方法确定了一个分布(包括参数的值)，进而我们探讨如何能产生具有这种分布的随机变量以便运行仿真模型。举例来说，如在第1.4节和第2章讨论的排队类型的模型，需要生成到达间隔时间和服务时间来驱动仿真按时间运行；而第1.5节的库存模型则需要在需求发生时随机生成需求量。

正如我们在本章将要看到的，从任意分布或随机过程中生成随机变量的每一种方法所需要的基本要素是一个独立同分布的$U(0, 1)$随机变量源。因此，本质上说必须要有一个可用的、统计意义下可靠的$U(0, 1)$随机数发生器。大多数计算机的安装以及仿真软件包都有一个方便的随机数发生器，但是其中一些(特别是对于版本较旧的)并不能很好地完成任务(参见第7章)。没有一个合格的随机数发生器，就不可能从任意分布中生成正确的随机变量。因此，在本章余下部分里，我们将假定有一个好随机数源可用。

通常有多种算法可以用来从给定的分布中生成随机变量。在特定仿真应用中，选择哪一算法来生成随机变量，应该考虑多种因素。但是，遗憾的是，这些因素通常都会相互冲突，因此，仿真分析人员使用哪一种算法的决定必然要做许多权衡。我们这里能做的是提出一些相关的问题。

第一个问题是准确性

我们认为，如果可能，所用算法应该能准确生成所需分布的随机变量，当然是在无法避免的计算机精度和$U(0, 1)$随机数发生器的准确性的外部限制范围内。目前对常用的各种分布都有正确而有效的算法，不必考虑任何老的近似方法(许多这种近似方法基于中心极限定理，例如，众所周知的得到“正态”随机变量的方法是用6个$U(0, 1)$随机变量之和，小于12个之和)。另一方面，很多实际工作人员认为，给定的分布无论如何只是一个对实际的近似，因此近似产生方法应该够了。但由于这依赖于具体情况且很难量化，所以我们仍推荐使用准确的生成方法。

第二个问题是效率

当我们能在多种准确生成方法中选择的时候，显然我们应倾向于使用高效率的算法，既包括存储空间，也包括运算时间。某些算法需要存储大量的常量或很大的数据表格，这可能被认为是较为棘手的或者至少是不方便的。至于执行时间，实际上有两个要素。首先，显然，我们希望能以少量的时间来完成每一个随机变量的产生，这个时间称为边际执行时间；其次，一些算法必须做一些初始运算，根据特定分布及参数来给定特定的常量或表，完成这项工作所需的时间称为准备时间。在大多数仿真运行中，我们将从给定分布生成大量的随机变量，因此边际执行时间恐怕比准备时间更为重要。但是，如果在仿真过程中分布的参数经常或是随机改变的，那么准备时间可能会变成需要重点考虑的因素。

第三个问题是复杂性

在选择一种算法中多少带有主观性的问题是算法的总体复杂性，包括概念上的以及执行上的因素。人们必然要问，用一种更复杂的算法可能带来的效率的提高是否值得花费巨大的努力来理解并实现它。对该问题的考虑与实现生成随机变量的方法的目的有关，一个效率更高但也更复杂的算法也许适用于固化到某一成熟的算法软件中，而并不适用于“一次性”的仿真模型。

第四个问题是技术

有一些问题属于技术性更多一些。某些算法依赖的随机变量源并不来自 $U(0, 1)$分布，这不是人们所希望的，尽管其他的事情是一样的。另一个技术问题是算法对某些参数值效率高，但对其他参数则代价大。我们希望算法对全部参数值都有高效率(有时称算法的鲁棒性，参见文献 Devroye(1988))。最后一个技术点是有关我们是否想使用某种方差减小技术来获取更好的(较小变化)估计(参见第 10、11、12 章)。常用的两种方差减小技术(公共随机数法和对偶变量法)需要在所研究的系统仿真中用到的基本 $U(0, 1)$输入随机变量实现同步，而这种同步对某些类型的随机变量生成算法来说更容易实现。特别是，通用的反变换法非常有助于实现所要求的同步和方差减小。第 8.2.1 小节将更准确地讨论这一点。

关于随机变量的产生的综合性的参考文献有很多，包括 Dagpunar(1988)、Devroye(1996)、Fishman(1996)、Gentle(2010)、Hormann(2004)和 Johnson(1987)。也有一些计算机软件包提供了很好的能力来生成范围很广的分布的随机变量，例如 IMSL 例程[Visual Numerics，Inc. (2013)]和 Press 等(2007)提供的 C 语言源代码(在第 7.2 和 7.3 节中)。

本章其他的内容组织如下：在第 8.2 节我们综述了随机变量生成最重要的通用方法，包括一些例子和不同方法的相关优缺点的一般讨论；在第 8.3 和 8.4 节我们给出了由仿真中非常有用的特定连续和离散分布产生随机变量生成算法。最后，在第 8.5 和 8.6 节我们讨论两个专门的主题：相关随机变量生成和平稳及非平稳到达过程的实现的生成。

8.2 产生随机变量的通用方法

有许多生成随机变量的技术，当然，所用的特定方法依赖于我们希望从哪一种分布中产生；但是，几乎所有这些技术都可根据其理论基础进行分类。在本节我们讨论这些通用方法。

8.2.1 反变换法

假定我们要产生一个连续随机变量 X(参见第 4.2 节)，X 有分布函数 F，当 $0<F(x)<1$ 时，F 是连续的且严格递增的(这意味着如果 $x_1<x_2$ 且 $0<F(x_1)\leqslant F(x_2)<1$，则事实上 $F(x_1)<F(x_2)$)。令 F^{-1}表示函数 F 的逆，则生成具有分布函数 F 的随机变量 X 的算法如下(回忆一下，～读作“分布为”)：

(1) 产生 $U\sim U(0, 1)$；

(2) 返回 $X=F^{-1}(U)$。

注意 $F^{-1}(U)$总是有定义的，因为 $0\leqslant U\leqslant 1$ 且 F 的范围是$[0, 1]$。图 8.1 所示的是该算法，其中与该分布函数相应的随机变量可以为正值或负值，特定的 U 值决定了它会是何种情形。在该图中，随机数 U_1 得到了正随机变量 X_1，而随机数 U_2 得到负的随机变量 X_2。

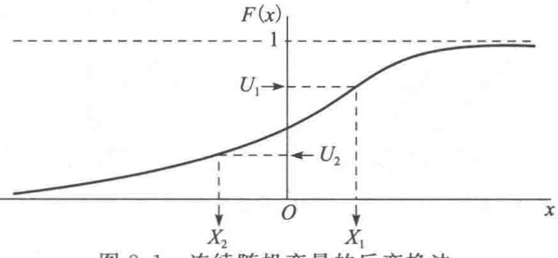


图 8.1 连续随机变量的反变换法

上述算法称为通用反变换法，为了证明由上述算法返回的值 X 具有所要求的分布 F，我们必须证明，对于任意实数 x，都有$P(X\leqslant x)=F(x)$。由于 F 是可逆的，我们有

$$P(X \leqslant x) = P(F^{-1}(U) \leqslant x)$$
$$= P(U \leqslant F(x)) = F(x)$$

其中，最后一个等式成立的原因是由于$U \sim U(0,1)$且$0<F(x)<1$(参见第6.2.2节中均匀分布的讨论)。

例8.1 令X具有均值为β的指数分布(参见第6.2.2小节)，其分布函数为：

$$F(x)=\begin{cases}1-e^{-x/\beta}, & x \geqslant 0 \\ 0, & \text{其他}\end{cases}$$

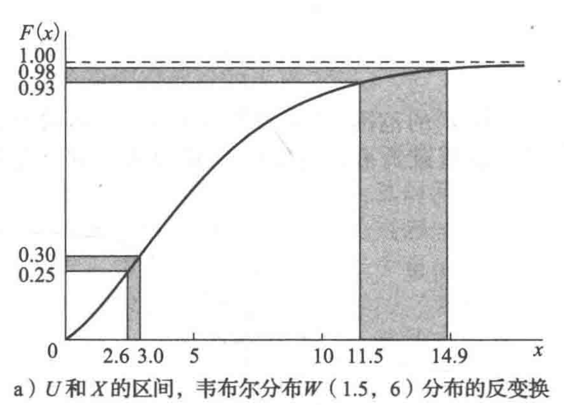
则为得到F^{-1}，我们设$u=F(x)$并求解x，以得到

$$F^{-1}(u) = -\beta\ln(1-u)$$

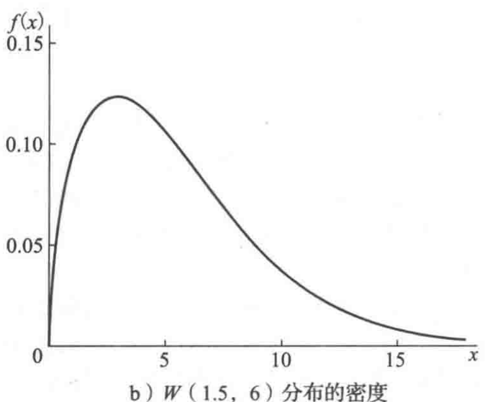
因此，要生成所要求的随机变量，我们首先生成一个$U \sim U(0,1)$，而后令$X=-\beta\ln U$(在该情形下可以使用U来代替$1-U$，是由于$1-U$和U有相同的$U(0,1)$分布，这节省一次减法运算)。 ◀

在上述例子中，我们为了略微提升算法效率用U来代替$1-U$。但是在像这样的情况下用U来代替$1-U$，会导致X与U负相关，而不是正相关。而且，正如在第8.3.15小节中看到的那样，并不是在变量生成算法中一旦出现$1-U$就都能由U来替换的。

反变换法在连续情形下的正确性上文已经在数学上进行了证明，此外，还很直观。一个连续随机变量的密度函数$f(x)$可以解释为在一个范围内的不同部分观测变量的相对机会。X轴上$f(x)$数值大的区域上我们将期望观测到的随机变量多，而在$f(x)$数值小的区域我们应该找到的随机变量只会很少。举例来说，在图8.2b表示的是韦布尔分布的概率密度函数，其形状参数$\alpha=1.5$，比例参数$\beta=6$(参见第6.2.2小节关于该分布的定义)，且我们能预期很多生成的随机变量会落在，比如说，$x=2$和$x=5$之间，没有太多落在$x=13$和$x=16$之间。图8.2a表示的是相应的分布函数$F(x)$。因为概率密度函数是分布函数的微分(即$f(x)=F'(x)$)，我们可以将$f(x)$看做$F(x)$的“斜率函数”，即$f(x)$是F在点x处的斜率。因此，对$f(x)$大的地方的x的值来说，F上升比较陡峭(例如对$x=2$和$x=5$之间)；相反，在$f(x)$小的区域，F则相对平坦(例如$x=13$和$x=16$之间)。那么，反变换法说，取随机数U，它应该平均地(均匀地)散布在$F(x)$图形的纵坐标的[0, 1]区间上，且是“直角拐弯的”，碰到$F(x)$陡处的U会多于平坦处的，从而将X集聚在$F(x)$陡处——准确地说是$f(x)$值高的区域。在图8.2a中，纵轴上的区间[0.25, 0.30]应该包含有5%的U，这导致X位于x轴上一个相对窄的区域[2.6, 3.0]内，因此大约5%的X将在此区域内。另一方面，纵轴区间[0.93, 0.98]，它与[0.25, 0.30]同样长度，因此也包含有大约5%的U，但导致X将位于x轴上一个大区域[11.5, 14.9]里。这里我们也会得到大约5%的X，但散布在大得多的区间上。



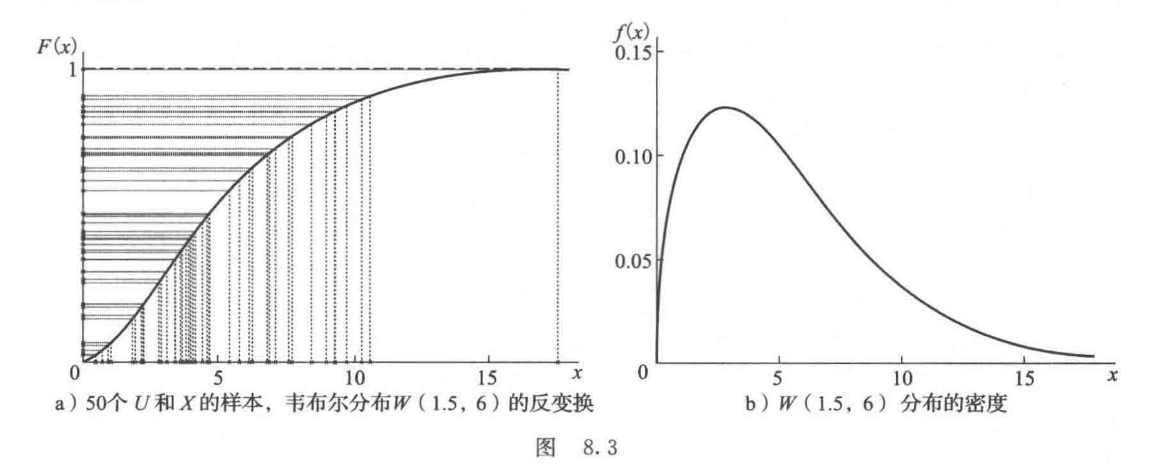
a) U和X的区间，韦布尔分布W(1.5, 6)分布的反变换



b) W(1.5, 6)分布的密度

图 8.2

图 8.3 表示了一个正在运行中的反变换法，生成了 50 个 X(用流 1 使用附录 7A 中的随机数发生器)。U 图示在图 8.3a 的纵轴上，与之对应的 X 按 U 的直角拐弯虚线得到。注意，纵轴上 U 的分布是相当均匀的，但在横轴上的 X 在密度函数 $f(x)$值高的区域的确较密，而在 $f(x)$值低处变得较疏。因此，反变换法本质上是将原有 U 的均匀分布按所要求的密度变形以得到 X 的分布。



a）50个 U 和 X 的样本，韦布尔分布 W（1.5，6）的反变换

b）W（1.5，6）分布的密度

图 8.3

反变换法也可用于 X 是离散的情形，此时分布函数为：

$$F(x) = P(X \leqslant x) = \sum_{x_i \leqslant x} p(x_i)$$

其中，$p(x_i)$是概率质量函数，

$$p(x_i) = P(X = x_i)$$

我们假定 X 只能取 x_1，x_2，…，其中，$x_1 < x_2 < \cdots$。则算法如下：

(1) 产生 $U \sim U(0, 1)$。

(2) 确定最小正整数 I 使得 $U \leqslant F(x_I)$并返回 $X = x_I$。

图 8.4 表示了该方法，其中我们产生了该情形下的 $X = x_4$。虽然该算法似乎与连续随机变量的反变换法没有关系，但图 8.1 和图 8.4 之间的相似性是显然的。

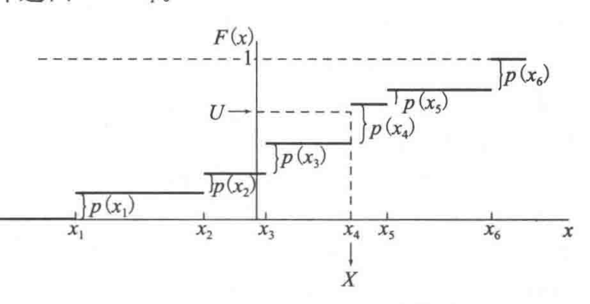


图 8.4 离散随机变量的反变换法

为了验证离散反变换法的有效性，我们需要证明对于所有 i，都有 $P(X = x_i) = p(x_i)$。对于 $i = 1$，当且仅当 $U \leqslant F(x_1) = p(x_1)$时，我们取 $X = x_1$，因为我们将 x_i 按增序排列了。由于 $U \sim U(0, 1)$，正如所要求的，$P(X = x_1) = p(x_1)$。对 $i \geqslant 2$，当且仅当 $F(x_{i-1}) < U \leqslant F(x_i)$时该算法置 $X = x_i$，因为由该算法确定的 i 是满足 $U \leqslant F(x_i)$的最小正整数。进一步，由于 $U \sim U(0, 1)$且 $0 \leqslant F(x_{i-1}) < F(x_i) \leqslant 1$，有

$$P(X = x_i) = P[F(x_{i-1}) < U \leqslant F(x_i)] = F(x_i) - F(x_{i-1}) = p(x_i)$$

例 8.2 回想第 1.5 节的库存例子，其中需求量随机变量 X 是离散的，取值 1，2，3 和 4 的概率分别为 $\frac{1}{6}$，$\frac{1}{3}$，$\frac{1}{3}$，$\frac{1}{6}$，其分布函数 F 在图 4.2 中给出。为产生 X，首先产生 $U \sim U(0, 1)$，根据 U 落到[0，1]的某一个子区间来确定 X 的值是 1，2，3 和 4 的哪一个。如果 $U \leqslant \frac{1}{6}$，则令 $X = 1$；如果 $\frac{1}{6} < U \leqslant \frac{1}{2}$，则令 $X = 2$；如果 $\frac{1}{2} < U \leqslant \frac{5}{6}$，则令 $X =$

3；最后，如果$\frac{5}{6}<U$，则令 $X=4$。◀

虽然图 8.4 和例 8.2 处理的都只是有限多个值的离散随机变量，但离散反变换法同样可直接用于产生所谓具有无限范围的随机变量，例如泊松分布、几何分布和负二项分布。

离散反变换法写成像例 8.2 中那样的方式的确是相当直观的。我们将单位区间分成宽度为 $p(x_1)$，$p(x_2)$，…的相邻子区间，并按照哪个子区间包含了所产生的 U 来对 X 赋值。例如，U 落在第二个子区间的概率是 $p(x_2)$，在这种情形下我们令 $X=x_2$。该算法的效率将取决于我们如何寻找包含给定 U 的子区间。最简单的方法是，从最左侧区间开始逐一寻找，首先检查 U 是否小于或等于 $p(x_1)$，如果是这种情况，我们返回 $X=x_1$；如果 $U>p(x_1)$，则检查是否 $U\leqslant p(x_1)+p(x_2)$，如果是这种情况，我们返回 $X=x_2$，等等。因此，确定 X 所需要做比较的次数将取决于 U 与 $p(x_i)$。例如，如果开头几个 $p(x_i)$都很小，那么在算法终止前我们必须做大量比较的概率就高。这就告诉我们进行这种搜索应以一种更复杂的方式，使用来自计算机科学文献的一种适当的分类和搜索技术(例如，参见文献 Knuth(1998b))。一个简单的改进便是首先检查 U 是否落在最宽的子区间中，因为这应该是一个最可能的情况。如果不是，我们应该检查第二宽的子区间，以此类推。当某些 $p(x_i)$的值明显大于其他的值因而 x_i 的个数多时，此方法应该特别有效，关于这一思想的更多内容请参见习题 8.2，也可参见文献 Chen，Asau(1974)和 Fishman，Moore(1984)，他们采用索引的思想给出了非常有效的搜索方法。

反变换法的通用性、优点以及缺点

反变换法的连续和离散版本两者可以结合在一起，至少在形式上，而成为更通用的形式：

$$X=\min\{x:F(x)\geqslant U\}$$

这种形式的附加优点是对于混合分布即分布同时包含连续和离散部分，以及具有平坦点的连续分布函数都是有效的。为了检查上式在连续情形的有效性，请注意图 8.1 中，集合$\{x:F(x)\geqslant U_1\}$是区间$[X_1, +\infty)$，其最小值为 X_1。在离散情形，我们在图 8.4 中看到$\{x:F(x)\geqslant U\}=[x_4, +\infty)$，其最小值为 x_4。图 8.5 表示一个有两个跳变的不连续点和一个平坦点的混合分布，在这种情形，有关的随机变量 X 应满足 $P(X=x_1)=u_1'-u_1$(在 x_1 的跳变)，$P(X=x_2)=u_2'-u_2$(在 x_2 的跳变)，以及 $P(x_0\leqslant X\leqslant x_0')=0$(在 x_0 与 x_0' 之间的平坦点)。对于其连续部分，注意

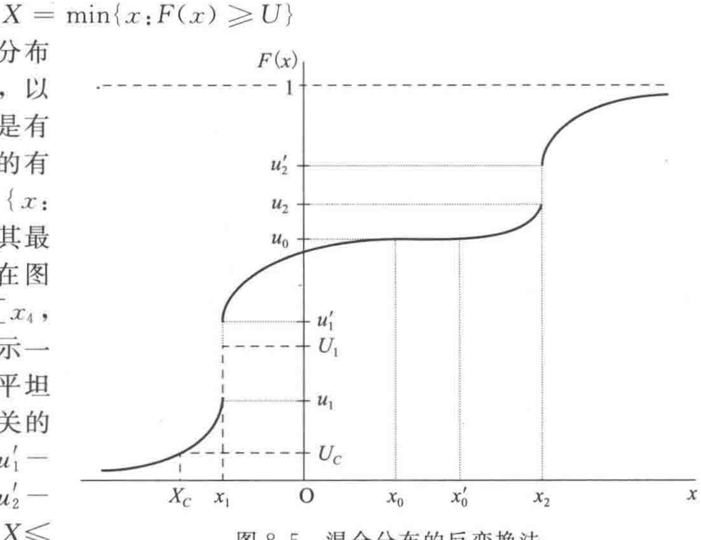


图 8.5 混合分布的反变换法

$$X=\min\{x:F(x)\geqslant U_C\}=\min[X_C,+\infty)=X_C$$

正如期望的那样。对于在 x_1 处的跳变不连续点，对于 $u_1\leqslant U\leqslant u_1'$，我们得到

$$X=\min\{x:F(x)\geqslant U_1\}=\min[x_1,+\infty)=x_1$$

它将以概率 $u_1'-u_1$ 发生，正如所希望的那样；在 x_2 处的跳变是类似的。对于平坦点，只有当我们产生的随机变量 U 等于 u_0 时，我们才会产生在(x_0, x_0')内的随机变量 X；由于 U 代表一个连续随机变量，所以 U 等于 u_0 发生的概率为零，尽管实际中产生的随机数 U 只有有限的精度，有可能会得到 $U=u_0$。因此，反变换法的这种更通用的描述适用于任何

连续、离散或混合分布的处理，尽管它实际上如何执行当然会很大程度上取决于所要求的分布形式。

下面，我们将讨论反变换法在连续和离散两种情况下的某些一般优点和缺点。在连续情况下应用这种方法一个可能出现的问题是需要计算 $F^{-1}(U)$。由于我们可能无法以闭合形式写出所要求的分布的 F^{-1} 的公式(例如正态分布和伽马分布)，因此，像例 8.1 那样简单的使用反变换法可能并不一定可行。但是，即使 F^{-1} 没有简单的闭合表达式，我们也可以使用如幂级数展开等的一些数值方法来计算 F^{-1}(例如，参见第 8.3 节有关产生伽马、正态和贝塔分布的随机变量的讨论)。这些数值方法可以达到任意精度，从而特别是可以与机器舍入误差所固有的精度相匹配；在这个意义上，对所有实际应用来说，这些方法是准确的。然而，Devroye(1986，第 31-35 页)指出，对于某些分布，特别是范围无限的分布，很难规定一个可接受的终止规则。Kennedy 和 Gentle(1980，第 5 章)对计算各类分布函数及其反函数的数值方法进行了详细的综述；也可参见 Abramowitz 和 Stegun(1964，第 26 章)和 Press 等(2007，第 6 章)。IMSL 算法库(Visual Numerics 公司，2013)中包含了计算大多数常见分布函数及其反函数的子例程，采用了仔细挑选过的算法。Marsaglia(1984)提出了另一种数值近似 F^{-1} 的算法，他建议要找到一个"接近" F^{-1} 又易于计算的函数 g()，然后使用 $g(Y)$ 来生成 X，其中，Y 是"接近" $U(0,1)$ 的特定分布。Marsaglia 将这个方法称为准确近似方法(也可参见 Fishman(1996，第 185-187 页))。

近年来，Hörmann 和 Leydold(2003)提出了一种在连续分布情形下的通用的适应性方法，以构建 F^{-1} 的高精度的 Hermite 插值。基于一次性的设置，此方法产生适度大小的插值点及系数的表格。然后使用这些表格可以非常快速地生成随机变量。该方法的代码可以在名为 UNURAN 的公共图书馆中找到。

反变换法第二个潜在缺点是，对于给定分布，反变换法或许不是生成相应随机变量的最快速方法，在第 8.3 与 8.4 节中，我们将对每个所考虑的分布讨论各种算法的效率。

虽然存在这些可能的缺陷，使用反变换法有一些重要的优点。第一个优点就是易于使用方差缩减技术，它依赖于引入随机变量间的相关性，该技术的例子是公共随机数法和对偶变量法。如果 F_1 和 F_2 是两个分布函数，则 $X_1=F_1^{-1}(U_1)$ 和 $X_2=F_2^{-1}(U_2)$ 是分别具有分布函数 F_1 和 F_2 的随机变量，其中 U_1 和 U_2 是随机数。如果 U_1 和 U_2 是独立的，则必然 X_1 和 X_2 也是独立的。但是，如果我们令 $U_1=U_2$，则使得 X_1 和 X_2 之间尽可能正相关，而取 $U_2=1-U_1$(回忆一下，它也是[0，1]上均匀分布的)使得 X_1 和 X_2 之间尽可能负相关。因此，反变换法能在生成的随机变量间引入最强的相关性(正相关或负相关)，我们希望的这种相关性将通过仿真模型传播，在其输出中引入最强可能的相关性，从而为方差缩减技术的成功做出贡献(当然，通过反变换法以外的其他方法也可以在随机变量中引入相关性，参见 Schmeiser 和 Kachitvichyanukul(1990))。在更注重实效的层面上，反变换法易于应用方差缩减技术，因为我们总是只需要一个随机数来产生一个所要求的 X 值(后面讨论的其他方法会需要多个随机数来生成一个 X 值，或者随机数的个数本身就是随机的，如在舍选法中的那样)。看到这一点非常重要，因为很多方差缩减技术的正确执行需要不同仿真运行之间的输入随机数能在一定程度上同步。如果使用反变换法，同步易于实现。

反变换法的第二个优点是，易于从截断分布中产生随机变量(参见第 6.8 节)。在连续的情况下，假设我们有密度函数 f，其相应的分布函数为 F。对 $a<b$(有可能 $a=-\infty$ 或 $b=+\infty$)，我们定义截断密度：

$$f^*(x)=\begin{cases}\dfrac{f(x)}{F(b)-F(a)}, & a\leqslant x\leqslant b\\ 0, & \text{其他}\end{cases}$$

它相应的截断分布函数为：

$$F^*(x)=\begin{cases}0, & x<a\\ \dfrac{F(x)-F(a)}{F(b)-F(a)}, & a\leqslant x\leqslant b\\ 1, & b<x\end{cases}$$

离散情形是类似的。则产生具有分布函数 F^* 的 X 的算法为：

(1) 产生 $U\sim U(0,1)$；

(2) 令 $V=F(a)+[F(b)-F(a)]U$；

(3) 返回 $X=F^{-1}(V)$。

我们将此算法的证明留作一个练习(见习题 8.3)，即证明由该算法所定义的 X 的确具有分布函数 F^*。注意，反变换法的思想实际上使用了两次：首先在第二步中将 V 均匀分布在 $F(a)$和 $F(b)$之间，然后在第三步中得到 X(另一种产生 X 的方法参见习题 8.3，而习题 8.4 是另一类型的截断，它得到的分布函数与 F^* 不同)。

最后，反变换法对生成顺序统计会相当有用。设 Y_1，Y_2，…，Y_n是独立同分布的，具有一般分布函数 F，且对于 $i=1, 2, \cdots, n$，$Y_{(i)}$表示第 i 位最小的 Y_j。回忆第 6 章，$Y_{(i)}$称为大小为 n 的样本中第 i 位的顺序统计(顺序统计在有关可靠性或生存时间的仿真中十分有用，此时某个具有构件的系统会失效。如果 Y_j 是第 j 个构件的生存时间，则 $Y_{(1)}$是由 n 个这样的构件串联构成的系统的生存时间，而 $Y_{(n)}$是如果所有构件并联时系统的生存时间)。一个直接产生 $X=Y_{(i)}$的方法是，首先产生 n 个分布函数为 F 的独立同分布变量 Y_1，Y_2，…，Y_n，之后将它们进行升序排列，最后置 X 为排序后的Y_j 第 i 位的值。但是，该方法需要产生 n 个具有分布函数 F 的独立的变量且将它们排序，如果 n 很大，这会很慢。作为一种替代方法，我们可使用下列算法来产生 $X=Y_{(i)}$：

(1) 产生 $V\sim\beta(i, n-i+1)$；

(2) 返回 $X=F^{-1}(V)$。

这种算法的有效性在习题 8.5 中来确定。注意，第一步需要从贝塔分布中产生，我们在后面的第 8.3.8 小节中讨论这一点。不需要排序，同时我们只需要计算一次 F^{-1}，n 很大或 F^{-1}计算很慢时此方法特别有优势。两个重要的特殊情况是产生 n 个 Y_j 中的最大值或最小值，其中第一步变得特别简单。对于最小值，$i=1$，则第 1 步的 V 可由 $V=1-U^{1/n}$来定义，其中 $U\sim U(0,1)$；对于最大值，$i=n$，从而我们可在第 1 步设置 $V=U^{1/n}$(这两个特殊情形的验证参见习题 8.5)。关于生成顺序统计的更多内容，请参考文献 Ramberg 和 Tadikamalla(1978)、Schmeiser(1978a，1978b)以及 Schucany(1972)。

8.2.2 组合法

组合法用于当我们希望由其产生的分布函数 F 可表示为其他分布函数 F_1，F_2，…的凸组合的时候。我们希望从 F_j 采样能比从原来的 F 采样容易得多。

特别是，我们假设，对于所有 x，$F(x)$可写为：

$$F(x)=\sum_{j=1}^{+\infty}p_jF_j(x)$$

其中，$p_j\geqslant 0$，$\sum_{j=1}^{+\infty}p_j=1$，且每一个 F_j 都是一个分布函数(虽然我们将此组合写成有无限多项的和，但可以存在一个 k，使得 $p_k>0$ 但对于 $j>k$，$p_j=0$，在此种情形下，实际上是有限项和)。同样，如果 X 的密度为 f，且可写成为：

$$f(x)=\sum_{j=1}^{\infty}p_jf_j(x)$$

其中 f_j 是别的密度。此时组合法仍然适用，离散情况是类似的。那么，通用的组合算法为：

(1) 产生一个正随机整数 J，满足

$p(J=j)=p_j$，对于 $j=1, 2, \cdots$

(2) 返回具有分布函数 F_j 的 X。

组合法的第一步可以看做是以概率 p_j 选择分布函数 F_j，例如使用离散反变换法可完成这一步。得到 $J=j$ 后，在第2步产生 X，当然要与 J 独立。根据在第一步产生 J 值的条件，我们可以很容易看出该算法返回的 X 将具有分布函数 F(例子参见 Ross(2003，第3章))：

$$P(X \leqslant x) = \sum_{j=1}^{+\infty} P(X \leqslant x \mid J = j) P(J = j) = \sum_{j=1}^{+\infty} F_j(x) p_j = F(x)$$

有时我们可以给出组合法的几何解释。举例来说，对于一个具有密度函数 f 的连续随机变量 X，我们可将 f 下的面积分成区域面积 p_1，p_2，…，对应于将 f 分解为凸组合表示。然后，我们可将步骤(1)视为选择一个区域，而步骤(2)视为由所选区域所对应的分布中产生。下述两个例子符合这种几何解释。

例 8.3 双指数分布(或拉普拉斯(Laplace)分布)的密度函数 $f(x) = 0.5e^{-|x|}$，x 为所有实数；图8.6画出了此密度函数的图形。由图我们可以看出，不考虑正则化因子0.5，$f(x)$是由两个"背对背"放置的指数密度函数；这就提示我们使用组合法。的确，我们可将此密度函数表示为：

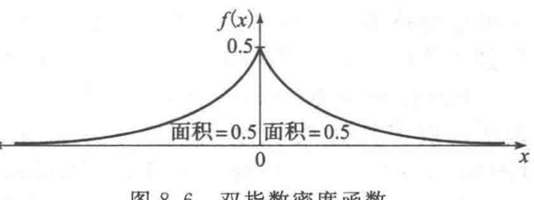


图 8.6　双指数密度函数

$$f(x) = 0.5e^{x} I_{(-\infty,0)}(x) + 0.5e^{-x} I_{[0,+\infty)}(x)$$

其中，I_A表示集合 A 的指示函数，定义为：

$$I_A(x) = \begin{cases} 1, & x \in A \\ 0, & \text{其他} \end{cases}$$

因此，$f(x)$是 $f_1(x)=e^x I_{(-\infty,0)}(x)$和 $f_2(x)=e^{-x} I_{[0,+\infty)}(x)$的凸组合，其中两者都是密度函数，而 $p_1=p_2=0.5$。由此我们可以使用组合法产生具有密度函数 f 的 X。首先产生 U_1 和 U_2 为 IID $U(0, 1)$；如果 $U_1 \leqslant 0.5$，则返回 $X=\ln U_2$；相反，若 $U_1 > 0.5$，则返回 $X=-\ln U_2$。注意，我们本质上是产生一个均值为1的指数分布的随机变量，而后以0.5的概率改变其符号。或者说，我们从图8.6的左半密度函数产生的概率等于其相应的面积(0.5)，且以0.5的概率从右半部分产生。◀

注意，在例8.3中，通用组合算法的第(2)步是使用了指数随机变量的反变换法来实现的，这表明生成随机变量的不同的通用方法是可以组合的。而且，我们看到在本例中需要两个随机数来生成一个 X；一般说来，我们将至少需要两个随机数来使用组合法(读者或许会发现，将例8.3与使用反变换法产生一个双指数随机变量相比较会很有趣，参见习题8.6)。

在例8.3中，我们通过用垂直线即坐标轴将密度函数下的区域分割开来以得到 f 的表达式。在下面的例子中，我们代之以进行水平分割。

例 8.4 对于 $0<a<1$，右梯形分布的密度函数可表示为：

$$f(x) = \begin{cases} a + 2(1-a)x, & 0 \leqslant x \leqslant 1 \\ 0, & \text{其他} \end{cases}$$

如图8.7所示，正如虚线所建议的，我们可考虑将 f 下的面积分为一个面积为 a 的矩形和一个面积为 $1-a$ 的右三角形。那么，$f(x)$可分解为：

$$f(x) = aI_{[0,1]}(x) + (1-a)2xI_{[0,1]}(x)$$

从而，$f_1(x)=I_{[0,1]}(x)$，其密度函数很简单为 $U(0, 1)$；$f_2(x)=2xI_{[0,1]}(x)$是右三角密度函数；显然，$p_1=a$，且 $p_2=1-a$。这

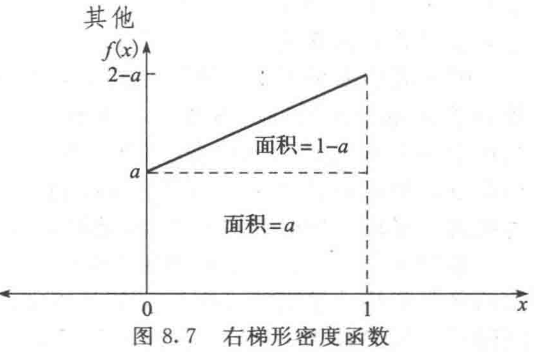


图 8.7　右梯形密度函数

样，组合法要求产生 $U_1 \sim U(0,1)$ 并检查是否 $U_1 \leqslant a$，若是，则产生一个独立的 $U_2 \sim U(0,1)$，并返回 $X=U_2$；但是，若 $U_1>a$，我们必须从右三角分布中产生。做到这一点有两种方法，一种是产生 $U_2 \sim U(0,1)$ 并返回 $X=\sqrt{U_2}$；或者产生分布为 IID $U(0,1)$ 的 U_2 和 U_3 并返回 $X=\max\{U_2, U_3\}$（参见习题 8.7）。由于取开方根运算的时间恐怕比多产生一个 $U(0,1)$ 的随机变量加上作一次比较的时间要多，因此第二种方法应是产生具有密度函数 f_2 的 X 的一种较快的方法。◀

还要说明的是，鼓励读者使用反变换法从例子 8.4 的右梯形分布中产生随机变量。但须注意，特别是当 a 很大时，组合法会快于反变换法，因为后者总要做开方运算，而前者有相当大的可能（以概率 a）只是简单地返回 $X=U_2 \sim U(0,1)$。分析人员要在速度的提升与必须产生两或三个随机数以得到一个 X 的值这样的缺点之间进行权衡。像例 8.4 中这样的梯形分布在开发高效方法中起到很重要的作用，例如 Schmeiser 和 Lal(1980) 开发了产生伽马随机变量的方法以及 Schmeiser 和 Babu(1980) 开发了产生贝塔随机变量的方法。

Peterson 和 Kronmal(1982) 对组合法（又称“混合”法）进行了更深入的研究。他们还说明，很多特定的变量发生方法实际上都可表示为某类的组合。其后，Peterson 和 Kronmal(1981，1982) 提出了与组合法紧密相关的一种有趣的方法——补选法（acceptance-complement method）；Devroye(1986，第 75-81 页) 对此及其相关方法进行了进一步讨论。

8.2.3 卷积法

对于一些重要的分布，所希望的随机变量 X 可表示为其他独立同分布的随机变量之和，而能比直接产生 X 要更容易地产生。我们假设有独立同分布的随机变量 $Y_1, Y_2, \cdots, Y_m$（m 为固定），使得 $Y_1+Y_2+\cdots+Y_m$ 与 X 有着相同的分布，由此我们写成：

$$X = Y_1 + Y_2 + \cdots + Y_m$$

该方法的名称，卷积法取自于随机过程中的术语，其中，X 的分布称为 Y_j 分布的 m 重卷积。读者应注意不要将这种情况与组合法混淆起来。在这里我们假定随机变量 X 可以表示成其他随机变量的和，而组合法后面的假设是 X 的分布函数可表示成其他分布函数的（加权）和，这两种情况有着本质的区别。

卷积法产生所要求的随机变量 X 的步骤是相当直观的（令 X 的分布函数为 F，Y_j 的分布函数为 G）：

(1) 产生为具有分布函数 G 的独立同分布随机变量 $Y_1, Y_2, \cdots, Y_m$。

(2) 返回 $X=Y_1+Y_2+\cdots+Y_m$。

为了说明该算法的有效性，回忆一下我们假设了 X 和 $Y_1+Y_2+\cdots+Y_m$ 有着相同的分布函数，即 F，这样

$$P(X \leqslant x) = P(Y_1 + Y_2 + \cdots + Y_m \leqslant x) = F(x)$$

例 8.5 均值为贝塔的 m 重厄兰分布的随机变量 X 可以定义为 m 个具有共同均值 β/m 的独立同分布的指数分布（均值为 β/m）的随机变量之和。因此，我们可以首先生成独立同分布的 $Y_1, Y_2, \cdots, Y_m$（参见例 8.1），之后生成 $X=Y_1+Y_2+\cdots+Y_m$（在第 8.3.3 小节中，我们将对此算法的效率作进一步的提升）。◀

倘若我们能易于产生随机变量 Y_j，当能使用卷积法时，它是非常简单的。但是，由于依赖于 X 的分布的特定参数，它并不一定会是最有效率的方法。举例来说，为了用卷积法产生一个 m 重厄兰随机变量（如在例 8.5 中那样），当 m 很大时，可能会很慢。在这种情况下，最好要想到 m 重厄兰分布是伽马分布的一个特殊情形（参见第 6.2.2 小节）而使用生成伽马随机变量的通用方法（参见第 8.3.4 小节）。也可参见 Devroye(1988)。

卷积法实际上是一个更通用思想的例子，将某些中间随机变量变换为具有所要求的分布的最终变量，卷积的变换仅仅是做加法，中间变量为独立同分布的。有很多其他变换中间随机变量的方法，其中一些将在第 8.2.6 小节、第 8.3 节和第 8.4 节中讨论。

8.2.4 舍选法

到目前为止讨论的产生随机变量的三种方法(反变换法、组合法与卷积法)可称为直接方法，其含义是它们都直接面对所要求的分布或随机变量。舍选法在方法上是一种并不这么直接的，因此可以用于在直接方法无效或效率不佳的情形。我们的讨论是面向连续情形的，其中，我们要产生具有分布函数 F 及密度函数 f 的 X。离散情形完全类似并在习题 8.9 中处理。其基本思想时间上至少可回溯到文献 Von Neumann(1951)。

舍选法需要我们指定一个函数 t，称为强函数，即对于所有 x，$t(x) \geqslant f(x)$。这样一来 t 一般来说就不是一个密度函数，因为

$$c = \int_{-\infty}^{+\infty} t(x)\mathrm{d}x \geqslant \int_{-\infty}^{+\infty} f(x)\mathrm{d}x = 1$$

但是，函数 $r(x) = t(x)/c$ 显然是一个密度函数(我们假定 t 是使 $c < +\infty$)。我们必须能产生(我们希望易行且快速)一个具有密度函数 r 的随机变量 Y。该通用的算法为：

(1) 产生具有密度函数 r 的 Y。

(2) 产生独立于随机变量 Y 的 $U \sim U(0, 1)$。

(3) 若 $U \leqslant f(U)/t(Y)$，则返回 $X = Y$。否则返回到步骤(1)重新再试。

算法会不停地循环返回到步骤(1)，直到我们在步骤(1)和(2)最终产生了满足 $U \leqslant f(U)/t(Y)$ 的一对随机变量(Y, U)，这时，我们将“接受”Y 的值赋给 X。由于说明该算法的有效性较前三种方法都复杂，因此我们建议读者参考附录 8A 中的证明。

例 8.6 $\beta(4, 3)$分布(在单位区间上)的密度函数为：

$$f(x) = \begin{cases} 60x^3(1-x)^2, & 0 \leqslant x \leqslant 1 \\ 0, & \text{其他} \end{cases}$$

由于其分布函数 $F(x)$是一个六阶多项式，因此反变换法不会简单，需要使用数值方法求多项式的根。使用标准的微积分，即设 $\mathrm{d}f/\mathrm{d}x = 0$，我们看到 $f(x)$的最大值发生在 $x = 0.6$处，此时 $f(0.6) = 2.0736$(准确值)。因此，如果我们定义：

$$t(x) = \begin{cases} 2.0736, & 0 \leqslant x \leqslant 1 \\ 0, & \text{其他} \end{cases}$$

则 t 是 f 的强函数。接下来，$c = \int_0^1 2.0736\mathrm{d}x = 2.0736$，所以 $r(x)$正好就是 $U(0, 1)$密度函数。函数 f，t 和 r 均在图 8.8 中给出。该算法首先在步骤(1)和(2)产生 Y 和 U 作为 IID $U(0, 1)$随机变量，之后在步骤(3)中我们检查是否满足：

$$U \leqslant \frac{60Y^3(1-Y)^2}{2.0736}$$

如果成立，我们返回 $X = Y$，否则，我们拒绝 Y 并返回步骤(1)。 ◀

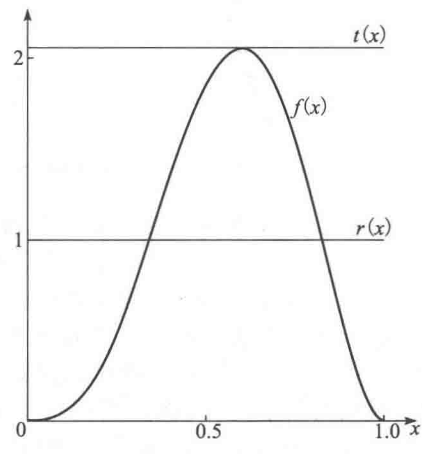


图 8.8　舍选法的 $r(x)$，$f(x)$和 $t(x)$，$\beta(4, 3)$分布

注意，在上述例子中，X 限定在在一个区间上(在该情形下是单位区间)，因此我们可以选择 t 为这个区间上的一个常数，从而使得 r 为均匀密度函数。舍选法经常说成只是用于限定区间的随机变量 X 以及只是用于选择 r 为均匀密度函数，我们的方法是更通用的。

上述舍选法至少说起来似乎很奇妙，而在附录 8A 中其有效性的证明将增加一些理解。然而，该方法有一个自然直观的表述。图 8.9 再次表示出了例 8.6 中的 $f(x)$和 $t(x)$的曲线，此外还给出该算法的运行过程。我们用舍选法由 $\beta(4, 3)$产生 50 个 X(使用附录 7A 中的随机数发生器的流 2)，它们用 x 轴上的叉标注出来。在图形顶端的 $t(x)$曲线上我们

标注了所有在该算法的步骤(1)中产生的所有 Y 的位置，无论它们最后是否被选为 X；这些 Y 中的 50 个被选取并将其下放到 x 轴上。$t(x)$曲线上的 Y 的均匀性是显而易见的，同时也能清晰看到，在 $f(x)$值大的 x 轴上的 X 有较高的集中性。对那些落入 $f(x)$值较小的区域的 Y 来说(例如 x 接近 0 或 1)，$f(Y)/t(Y)$小，而它是作为选取 Y 为 X 的概率，因此这些 Y 的大多数都会被舍弃。这在图 8.9 中能看到 $f(x)$小的地方，Y 值小(接近 0)或大(接近 1)。相反，$f(x)$大的地方，Y 值(例如接近 $x=0.6$)恐怕会保留，因为 $f(Y)/t(Y)$接近 1，因此，$x=0.6$ 附近的大多数 Y 都被接受为 X，并将其下放到 x 轴上。按这种办法，在 $t(x)$远大于 $f(x)$的地方该算法将 Y 从 $r(x)$密度函数中"剔除出来"，但保留 $t(x)$只比 $f(x)$稍大一点的地方的大多数 Y。其结果是改变了来自 $r(x)$的 Y 的集中性以满足所要求的密度函数 $f(x)$。

舍选法的原理是相当通用的，且从一个稍有不同的角度去观察上述算法就可弄清它如何能扩展到更高维空间中随机点的生成。例如，在多重积分的蒙特卡罗估计(参见第 1.8.3 小节)中，这是很重要的。舍选法的步骤 3 的选取条件显然可重新表述为 $U(t(Y))\leqslant f(Y)$，它的几何含义是，如果点$(Y, U(t(Y)))$落在密度函数 f 的曲线之下，Y 将被选取为一个 X。图 8.10 表示了与图 8.9 相同的 Y 值，用圆点表示点$(Y, U(t(Y)))$，被选取的 50 个 X 也用 x 轴上的叉加以标出。通过选取那些落在 $f(x)$曲线下方的点$(Y, U(t(Y)))$的 Y 值，可以很直观的看出，在 x 轴上 $f(x)$值高的地方被选的 X 更密集，这是由于均匀分布的点在这些区域将更容易落在 $f(x)$下方。虽然在这个特定例子中 $t(x)$下方的区域的矩形性质使得点$(Y, U(t(Y)))$的均匀性非常明显，但对于任何形状区域以及在任何维数中其真实性是相同的。挑战在于找到一种方法在任意非矩形区域中有效地产生均匀分布的点。Smith (1984)讨论了这个问题，并提出了一个在高维数空间中更为有效的舍选法的替代算法。

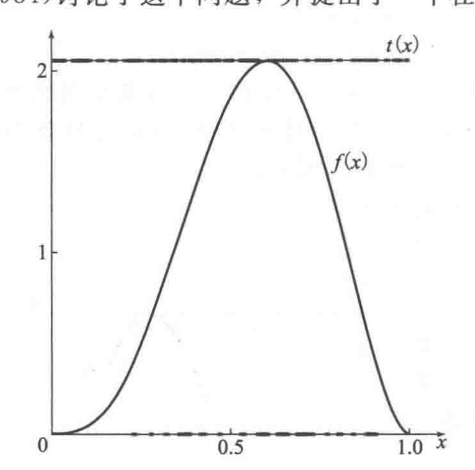


图 8.9 50 个 X 样本(水平轴上)以及所需要的 Y($t(x)$线上)对 $\beta(4, 3)$分布的舍选法

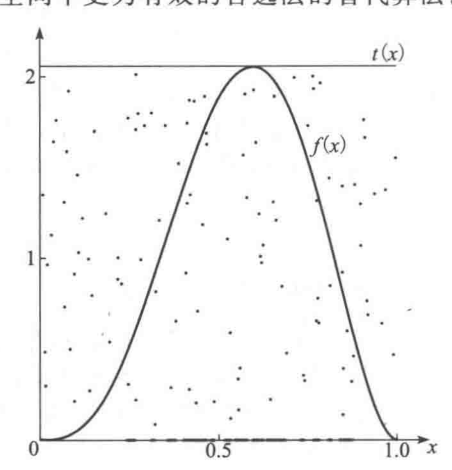


图 8.10 50 个 X 样本(水平轴上)以及所需要的$(Y, U(t(Y)))$对 $\beta(4, 3)$分布的舍选法

虽然无论强函数 t 如何选择舍选法会产生具有所要求分布的 X 的值，但是，选取的强函数 t 对舍选法的效率有两方面重要的影响：首先，在步骤(1)中需要产生具有密度函数 $t(x)/c$的 Y，我们希望选择的函数 t 能很快速地完成此步骤(例 8.6 中选择的均匀分布 t 肯定满足此要求)；其次，我们希望步骤(3)中舍弃的概率要小，因为如果该舍弃发生的话我们必须从头开始。在附录 8A 中我们证明了在整个算法中对于任意给定的迭代，步骤(3)中选取概率都是 $1/c$；因此我们希望选择 t 使得 c 比较小。也就是说，我们希望能找到一个紧贴 f 上面的 t，使得 c 更接近于其下界 1。直观地看，只在 f 上面一点点的 t 使得密度函数 r 会接近于 f。这样步骤(1)中从 r 中产生的 Y 值来自几乎正确的分布，从而我们应选取其绝大部分(那么，从这个观点出发，我们看到例 8.6 中 t 的选择并不是太理想，因为它并未非常紧密地下贴到 f 的顶部。因为 $c=2.0736$，选取的概率只有大约 0.48，低于我们的期望)。易于从 $t(x)/c$ 产生和一个小的 c 值这两个目标也许会有冲突，因此，t 的选择

绝不是显而易见的且值得注意。很多研究的目标是对给定的分布找到 t 的一个好选择，例如，请参见文献 Ahrens 和 Dieter（1972，1974）、Atkinson（1979b）、Atkinson 和 Whittaker(1976)、Schmeiser(1980a，1980b)、Schmeiser 和 Babu(1980)、Schmeiser 和 Lal(1980)、Schmeiser 和 Shalaby(1980)、Tadikamalla(1978)。寻找合适的 t 的一种流行的方法是：首先指定 $r(x)$ 为某一常见密度，例如正态分布或双指数分布，之后找到最小的 c 使得对于所有 x，满足 $t(x)=cr(x)\geqslant f(x)$。

例 8.7 再次考虑例 8.6 中的 $\beta(4，3)$分布，但现在使用更精细的强函数，力图在不过度增加从 $r(x)$ 产生 Y 的负担情况下提升选取概率；我们将按 Schmeiser 和 Shalaby (1980)的路线来做此事。对于此密度函数，它本身有两个拐点(即，$f(x)$在该 x 值点上从凸变凹，或者相反)，这些点可对 0 与 1 之间的 x 值求解 $f''(x)=0$ 得到，解是 $x=0.36$ 与 $x=0.84$(精确到 2 位十进制小数)。两个拐点将[0，1]分成了三个区域，分别判断三个区域上 $f''(x)$的符号，我们得到 $f(x)$在[0，0.36]上为凸，在[0.36，0.84]上为凹，且在[0.84，1]再次为凸。根据凸性的定义，从点(0，0)到点(0.36，$f(0.36)$)间的直线位于 $f(x)$之上；类似地，(0.84，$f(0.84)$)与点(1，0)间的连接线也位于 $f(x)$之上。在凹区域，我们只在 $f(x)$的最大值 2.0736 的高度放置一条水平线。这便得到了如图 8.11 所示的分段线性强函数 $t(x)$；将两个三角形区域和一个矩形区域相加，我们得到 $c=1.28$，从而按照给定的路径通过该算法选取概率是 0.78，比在例 8.6 中使用简单均匀分布的强函数的选取概率 0.48 要好得多。但是，现在从 $r(x)$产生值变得困难得多了，图 8.12 给出了 $r(x)$的图形；在确定一个强函数时这是一个典型的折中方法。正如图 8.12 中所建议的，从 $r(x)$产生可使用组合法，将 $r(x)$分为三个区域，每一个相应的密度函数都易于由其产生，参见习题 8.16。因此，我们这里实际上将三种不同的产生方法组合在一起：反变换法(用于 $r(x)$的部分密度函数)，组合法(用于 $r(x)$)以及最后的舍选法(用于 $f(x)$)。与例 8.6 相比较，为获得较高的选取概率而增加产生 Y 的工作量是否值得，这并不清楚，且会与若干因素有关，例如特定的参数、编码的效率，以及编程的语言、编译器和硬件。

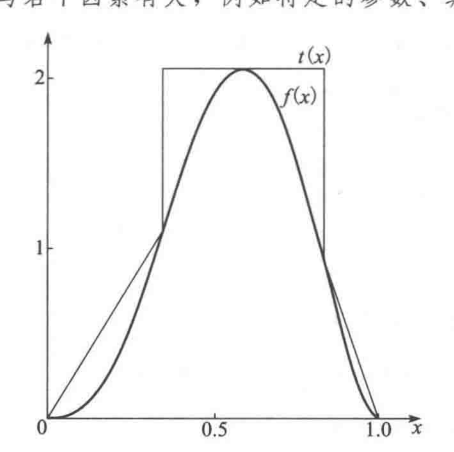


图 8.11　$f(x)$与分段线性强函数 $t(x)$的 $\beta(4，3)$分布的舍选法

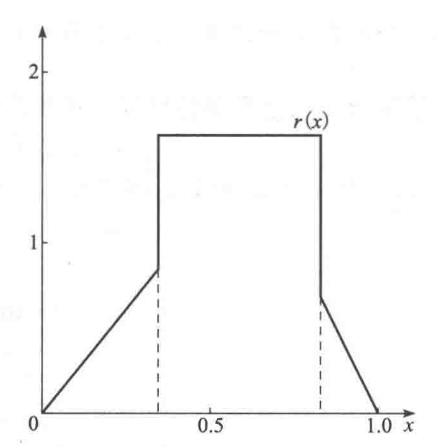


图 8.12　与图 8.11 所示的分段线性强函数 $t(x)$相对应的 $r(x)$ ◀

通用舍选法有很多变形和改进，主要是为了改进速度。例如，Hörmann 和 Derflinger (1996)给出了一个由离散分布(包括无穷大的尾)的产生的版本，它使用了连续的强函数以避免为确定是选取还是舍弃而生成单独的随机数。

8.2.5 均匀比法

令 $f(x)$是与连续随机变量 X 相应的概率密度函数，我们希望由它产生随机变量。均匀比法是由 Kinderman 和 Monahan(1977)提出的，它基于随机变量 U、V 和 V/U 之间的

奇妙关系。令 p 是一个正实数，如果(U, V)在集合 S

$$S=\left\{(u,v):0\leqslant u\leqslant\sqrt{pf\left(\frac{v}{u}\right)}\right\}$$

上是均匀分布的，则 V/U 的概率密度函数为 f。下面我们将证明它。U 和 V 的联合密度函数为：

$$f_{U,V}(u,v)=\frac{1}{s},(u,v)\in S$$

其中，s 是集合 S 的面积。

令 $Y=U$ 且 $Z=V/U$，该变换的雅可比(Jacobi)矩阵 $\boldsymbol{J}$ 为：

$$\boldsymbol{J}=\begin{vmatrix}\dfrac{\partial u}{\partial y} & \dfrac{\partial u}{\partial z}\\ \dfrac{\partial v}{\partial y} & \dfrac{\partial u}{\partial z}\end{vmatrix}=\begin{vmatrix}1 & 0\\ z & y\end{vmatrix}=y$$

因此，Y 和 Z 的联合分布为(参见文献，例如，DeGroot(1975，第 133-136 页))：

$$f_{Y,Z}(y,z)=|\boldsymbol{J}|f_{U,V}(u,v)=\frac{y}{s},\ 0\leqslant y\leqslant\sqrt{pf(z)}\text{ 且 }0<z<+\infty$$

从而，Z 的密度函数为：

$$f_Z(z)=\int_0^{\sqrt{pf(z)}}f_{Y,Z}(y,z)\mathrm{d}y=\int_0^{\sqrt{pf(z)}}\frac{y}{s}\mathrm{d}y=\frac{p}{2s}f(z)$$

由于 $f_Z(z)$和 $f(z)$的积分都必须为 1，因此有 $s=p/2$ 且 $Z=V/U$ 具有我们所要求的密度函数 $f(x)$。

要在 S 中均匀产生(u, v)，我们可以选择一个包含 S 的强区域 T，在 T 中均匀地产生(u, v)，并当满足下述条件时选取该点：

$$u^2\leqslant pf\left(\frac{v}{u}\right)$$

否则，我们在 T 中产生一个新点并再试，以此类推。希望在强区域 T 中应该易于均匀地产生一个点。

选取域 S 的边界条件可按下式参数化定义(参见习题 8.19)：

$$u(z)=\sqrt{pf(z)},v(z)=z\sqrt{pf(z)}\tag{8.1}$$

如果 $f(x)$和 $x^2f(x)$有界，则 T 的一个好选择是矩形：

$$T=\{(u,v):0\leqslant u\leqslant u^*\text{ 且 }v_*\leqslant v\leqslant v^*\}$$

其中，

$$u^*=\sup_z u(z)=\sup_z\sqrt{pf(z)}$$

$$v_*=\inf_z v(z)=\inf_z z\sqrt{pf(z)}$$

$$v^*=\sup_z v(z)=\sup_z z\sqrt{pf(z)}$$

集合(0, 1)的 sup(上确界)为 1，但其最大值不存在，inf(下确界)的定义是类似的。这样选择强区域 T 后，均匀比法正式描述如下：

(1) 独立地产生 $U\sim U(0, u^*)$与 $V\sim U(v^*, v^*)$；

(2) 置 $Z=V/U$；

(3) 如果 $U^2\leqslant pf(Z)$，则返回 Z，否则返回到步骤(1)。

如果 t 为区域 T 的面积，则选取一个特定 Z 的概率为：

$$\frac{s}{t}=\frac{p/2}{u^*(v^*-v_*)}$$

且通过步骤(1)到步骤(3)直到一个 Z 被选取平均次数为上述比率的倒数为止。

例 8.8 假设对于 $0\leqslant x\leqslant 1$，$f(x)=3x^2$（$\beta(3, 1)$分布，参见第6.2.2小节），且 $p=1/3$，则

$$S=\left\{(u,v):0\leqslant u\leqslant \frac{v}{v},0\leqslant \frac{v}{u}\leqslant 1\right\}$$

$$s=\frac{1}{6}$$

$$u^*=1,v_*=0,v^*=1$$

我们在正方形 T（面积 $t=1$）中均匀地产生2 000个点（u，v），其中330个点满足在 S 的定义中所说明的停止规则。注意，330/2 000=0.165，近似等于 $s/t=\frac{1}{6}/1=0.167$。这330个被选取的点的图形如图8.13所示。注意，选取区域 S 的上下界分别为曲线 $v=u$ 和 $v=u^2$（参见习题8.20）。◀

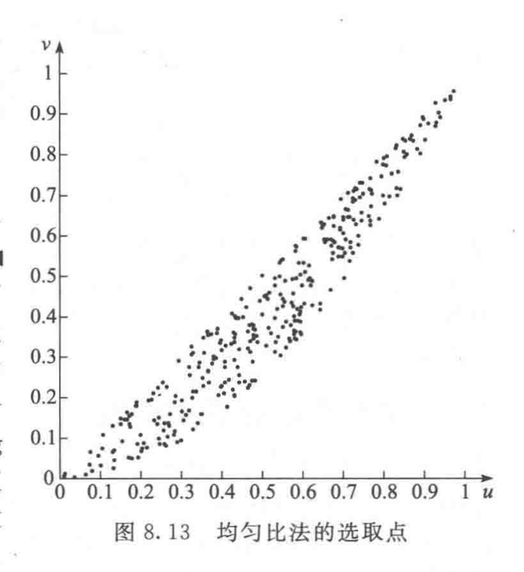


图8.13 均匀比法的选取点

例8.8中矩形 T 的选取概率为1/6。我们可以选择一个更接近选取域 S 的强区域 T，使得比率 s/t 会更接近于1（参见习题8.21）。但是，这种效率的提升与在更复杂的强区域均匀地产生一个点会有更大的难度，两者必须折中考虑。Cheng和Feast(1979)给出了一个从伽马分布产生的快速算法，此时 T 是平行四边形；Leydold(2000)开发了一种快速均匀比算法，算法使用多边形强区域，因而可用于一大类分布。

Stadlober(1990)证明了均匀比法实际上就是一种舍选法。他还将均匀比法拓展到了离散分布。同时Wakefield等(1991)与Stefănescu和Văduva(1987)还讨论了多变量分布。

8.2.6 特性法

虽然大多数产生随机变量的方法都可归类为在第8.2节到目前为止所讨论的五种方法之一，有一些技术却仅仅依赖于所要求的分布函数 F 或随机变量 X 的某些特性。特性往往会用更易于生成的其他随机变量来表示 X 的形式。在这个意义上，卷积是一种“特殊的”特性。下面的四个例子是基于正态理论随机变量的（对不能用正态分布来做的其他例子请参见第8.3和8.4节）。

例 8.9 如果 $Y\sim N(0, 1)$（标准正态分布），则 Y^2 具有自由度为1的 χ^2 分布（我们写 $X\sim\chi_k^2$ 的含义是 X 具有自由度为 k 的 χ^2 分布）。因此，要产生 $X\sim\chi_1^2$，先产生 $Y\sim N(0, 1)$（参见第8.3.6小节），并返回 $X\sim Y^2$。◀

例 8.10 如果 Z_1，Z_2，…，Z_k 是IID χ_1^2 随机变量，则 $X=Z_1+Z_2+\cdots+Z_k\sim\chi_k^2$。因此，要产生 $X\sim\chi_k^2$，首先产生 Y_1，Y_2，…，Y_k 作为IID $N(0, 1)$随机变量，之后返回 $X=Y_1^2+Y_2^2+\cdots+Y_k^2$（参见例8.9）。由于对于大的 k，此算法可能相当满，我们可使用如下事实：χ_k^2 分布是形状参数为 $\alpha=k/2$ 且比例参数 $\beta=2$ 的伽马分布。所以 X 可直接由第8.3.4小节中讨论的产生伽马分布的方法得到。◀

例 8.11 如果 $Y\sim N(0, 1)$，$Z\sim\chi_k^2$，且 Y 和 Z 独立，则 $X=Y/\sqrt{Z/k}$ 称作具有 k 自由度的学生分布，我们将它记为 $X\sim t_k$。因此，要生成 $X\sim t_k$，我们产生 $Y\sim N(0, 1)$ 并独立于 Y 产生和 $Z\sim\chi_k^2$（参见例8.10），之后返回 $X=Y/\sqrt{Z/k}$。◀

例 8.12 如果 $Z_1\sim\chi_{k_1}^2$，$Z_2\sim\chi_{k_2}^2$，且 Z_1 与 Z_2 独立，则

$$X = \frac{Z_1/k_1}{Z_2/k_2}$$

称为具有(k_1，k_2)自由度的 F 分布，记为 $X \sim F_{k_1,k_2}$。我们因此独立地产生 $Z_1 \sim \chi^2_{k_1}$，$Z_2 \sim \chi^2_{k_2}$，并返回 $X=(Z_1/k_1)/(Z_2/k_2)$。◀

对于某些连续分布，有可能对其概率密度函数进行变换，以使得易于构造用于舍选法的强函数。特别是，Hörmann(1995)提出了一种从密度函数为 f 的连续分布中产生随机变量的变换密度舍弃法。这种方法的思想是利用一个严格增函数 T 来变换 f，使得 $T(f(x))$ 为凹函数，在这种情况下我们将 f 称为 T-凹函数(对于 $x_1<x_2$，如果

$$\frac{g(x_1)+g(x_2)}{2} < g\left(\frac{x_1+x_2}{2}\right)$$

函数 g 称为是凹的)。由于 $T(f(x))$是一个凹函数，$T(f(x))$的强函数能易于构建为几个正切的最小值。然后用 T^{-1}将强函数变回到原来比例。这就得到密度函数 f 的强函数，并能用舍选法从 f 产生随机变量。如果 $T(x)=-1/\sqrt{x}$，则很多分布都是 T-凹的，包括贝塔分布、指数分布、伽马分布、对数正态分布、正态分布和韦布尔分布(对于某些分布，其参数值有约束)。由于变换密度舍弃法能用于一大类分布，它有时也称为一种万能方法(参见 Hormann(2004))。

8.3 连续随机变量的产生

在这一节中我们讨论从几种常见连续分布产生随机变量的特定算法；第 8.4 节包含离散随机变量的类似处理方法。虽然从一种给定分布产生可以有几种不同的算法，但我们对每一种情形只明确地给出一种技术，而对于其他算法则提供参考文献，在某种意义下，例如在以增加准备开销和更大的复杂度为代价的速度方面，这些算法也许更好。在决定给出哪种算法时，我们力图选择那些易于表述和执行，同时又有合理效率的算法。我们也只给出准确的(至多为机器精度)方法，并不给出近似算法。但是，如果速度是最重要的，我们希望读者去阅读给出的所要求的分布的各种参考文献。对于密度函数、质量函数和分布函数的定义，请参见第 6.2.2 小节和第 6.2.3 节。

8.3.1 均匀分布

$U(a, b)$随机变量的分布函数易于反变换，通过对 $0 \leqslant u \leqslant 1$ 求解 $u=F(x)$得到

$$x = F^{-1}(u) = a + (b-a)u$$

因此，我们可以使用反变换法来产生 X：

(1) 产生 $U \sim U(0, 1)$；

(2) 返回 $X=a+(b-a)U$。

如果有很多个 X 需要产生，那么常数 $b-a$ 应该预先计算并存储，以备算法使用。

8.3.2 指数分布

在例 8.1 中已经讨论过均值 $\beta>0$ 的指数分布随机变量，在那里我们得出了如下反变换算法：

(1) 产生 $U \sim U(0, 1)$；

(2) 返回 $X=-\beta \ln U$。

回忆一下，如果希望使用字面版本的 $X=F^{-1}(U)$以便使得 X 和 U 正相关，则步骤(2)中的 U 应该代之以 $1-U$。这一算法肯定非常简单，且拥有我们在第 8.2.1 小节中讨论的反变换法的全部优点。此算法还相当快，其主要的计算时间主要用于计算对数。在 Ahrens 和 Dieter(1972)的实验中，如果用 FORTRAN 编程，此算法在所考虑的四种算法中是最快的；算法 72%的时间花费在对数计算上。如果人们希望使用更低级的语言编程，有一些其他方法避免对数计算因而更快，但明显地更为复杂并包括不同数量的初始化准

备，参见 Von Neumann(1951)、Marsaglia(1961)、MacLaren、Marsaglia 和 Bray(1964)。对于进一步的讨论，我们推荐感兴趣的读者阅读文献 Ahrens 和 Dieter(1972)、Fishman (1978，第 402-410 页)。

8.3.3　m 厄兰分布

正如在例 8.5 中所讨论的那样，若 X 是一个均值为 β 的 m 厄兰随机变量，我们可写成为 $X=Y_1+Y_2+\cdots+Y_m$，其中，Y_i 是 IID 指数随机变量，每个均值为 β/m。这就得到在例 8.5 中所描述的卷积法生成 X。但是，这种算法的效率可以进一步提高，说明如下。如果我们使用第 8.3.2 小节中提到的反变换法来产生指数随机变量 Y_i($Y_i=(-\beta/m)\ln U_i$，其中，U_1，U_2，…，U_m 是 IID $U(0,1)$随机变量)，则：

$$X=\sum_{i=1}^{m}Y_i=\sum_{i=1}^{m}\frac{-\beta}{m}\ln U_i=\frac{-\beta}{m}\ln\left(\prod_{i=1}^{m}U_i\right)$$

这样我们只需计算一次对数(而非 m 次对数运算)。从而算法的描述如下：

(1) 产生 U_1，U_2，…，U_m 为 IID $U(0,1)$；

(2) 返回 $X=\frac{-\beta}{m}\ln\left(\prod_{i=1}^{m}U_i\right)$。

同样，人们应预先计算 β/m 并将其存储起来以备重复使用。此算法实际上是组合法与反变换法的结合。

由于我们必须产生 m 个随机数并执行 m 次乘法，因此算法的执行时间近似正比于 m。因此，当 m 值很大时人们可能会寻找其他算法。幸运的是，m 厄兰分布是伽马分布的一种特殊情形(形状函数 α 等于整数 m)，因此我们这里也可以使用一种产生伽马分布随机变数的方法(参见第 8.3.4 小节关于伽马分布的讨论)。应切换到通用的伽马分布产生的 m 的阈值取决于产生伽马随机变量所用的方法，以及语言、编译器和硬件环境等；对于特定的环境，进行初步实验证明是值得的(对于第 8.3.4 节的伽马发生器，在 $\alpha>1$ 的情况下，Cheng(1977)的计时实验表明，当大约 $m\geqslant 10$ 时，使用他的伽马发生器将比上述的 m 厄兰算法方法快)。使用上述算法的另一个潜在问题是，特别是当 m 很大的时候，$\prod_{i=1}^{m}U_i$ 可能会接近零，这会对之后取对数造成数值计算困难。

8.3.4　伽马分布

产生一般伽马随机变量要比本节到目前为止所讨论的三种随机变数要更复杂，因为其分布函数没有我们能做反变换的简单的闭合形式。首先要注意，对于给定的 $X\sim\Gamma(\alpha,1)$，对于任意 $\beta>0$，令 $X'=\beta X$，我们可得到 $\Gamma(\alpha,\beta)$随机变量 X'，从而，将注意力限制在 $\Gamma(\alpha,1)$分布就够了。进一步，回忆一下，$\Gamma(1,1)$分布恰好是均值为 1 的指数分布，因此我们只需考虑 $0<\alpha<1$ 和 $\alpha>1$。由于可用于产生伽马随机变量的算法一般都只对 α 的上述范围中的一种是最有效的，因此我们将分别讨论它们(Tadikamalla 和 Johnson(1981)给出了在当时可用的伽马随机变量产生方法的综述)。

我们首先考虑 $0<\alpha<1$ 的情况(注意，如果 $\alpha=0.5$，我们有一个缩放的 χ_1^2 分布且 X 可易于依例 8.9 的方法产生；同时下面所述的算法对 $\alpha=0.5$ 仍然有效)。Atkinson 和 Pearce (1976)测试了此情形下的三种算法，我们给出其中之一，这应归功于 Ahrens 和 Dieter (1974)(Forsythe(1972)的算法与 Atkinson 和 Pearce(1976)相比通常是最快的，但它复杂得多)。Ahrens 和 Dieter 在 1974 年提出的算法记为 G-S，是一种舍选法，其强函数为：

$$t(x)=\begin{cases}0, & x\leqslant 0\\ \dfrac{x^{\alpha-1}}{\Gamma(\alpha)}, & 0<x\leqslant 1\\ \dfrac{e^{-x}}{\Gamma(\alpha)}, & 1<x\end{cases}$$

因此，$c=\int_0^{+\infty} t(x)\mathrm{d}x = b/[\alpha\Gamma(\alpha)]$，其中 $b=[(e+\alpha)/e]>1$，由此可得密度函数 $r(x)=t(x)/c$为

$$r(x)=\begin{cases}0, & x\leqslant 0\\ \dfrac{\alpha x^{\alpha-1}}{b}, & 0<x\leqslant 1\\ \dfrac{\alpha \mathrm{e}^{-x}}{b}, & 1<x\end{cases}$$

使用反变换法可产生一个密度函数为 $r(x)$的随机变量 Y，相应于 r 的分布函数为：

$$R(x)=\int_0^x r(y)\mathrm{d}y=\begin{cases}\dfrac{x^{\alpha}}{b}, & 0\leqslant x\leqslant 1\\ 1-\dfrac{\alpha \mathrm{e}^{-x}}{b}, & 1<x\end{cases}$$

对上式可进行反变换，得到：

$$R^{-1}(u)=\begin{cases}(bu)^{1/\alpha}, & u\leqslant \dfrac{1}{b}\\ -\ln\dfrac{b(1-u)}{\alpha}, & \text{其他}\end{cases}$$

因此，要生成一个具有密度函数 r 的 Y，我们首先产生 $U_1\sim U(0,1)$。如果 $U_1\leqslant 1/b$，我们置 $Y=(bU_1)^{1/\alpha}$，此时 $Y\leqslant 1$。否则，如果 $U_1>1/b$，置 $Y=-\ln[b(1-U_1)/\alpha]$，将有 Y 大于 1。注意到：

$$\frac{f(Y)}{t(Y)}=\begin{cases}\mathrm{e}^{-Y}, & 0\leqslant Y\leqslant 1\\ Y^{\alpha-1}, & 1<Y\end{cases}$$

我们得到最终的算法(必须事先计算 $b=(e+\alpha)/e$)：

(1) 产生 $U_1\sim U(0,1)$，并令 $P=bU_1$，如果 $P>1$，转到步骤(3)，否则推进到步骤(2)；

(2) 令 $Y=P^{1/\alpha}$，并产生 $U_2\sim U(0,1)$，如果 $U_2\leqslant \mathrm{e}^{-Y}$，则返回 $X=Y$，否则返回到步骤(1)；

(3) 令 $Y=-\ln[(b-P)/\alpha]$，并产生 $U_2\sim U(0,1)$，如果 $U_2\leqslant Y^{\alpha-1}$，则返回 $X=Y$，否则返回到步骤(1)。

我们现在来讨论 $\alpha>1$ 的情况，这里有若干优秀的算法。根据 Schmeiser 和 Lal (1980)、Cheng 和 Feast(1979)给出的计时实验的观点，我们将给出一种由 Cheng(1977)提出的改进舍选法，他将此方法称作 G-B 算法。该算法有一个“帽型”的执行时间，也就是说它的执行时间在 $\alpha\to+\infty$时是有界的，并且事实上随着 α 的增大而变快(这种对通用舍选法的改进包括增加了一个对选取的快速预测试)。为了获得强函数 $t(x)$，首先令 $\lambda=\sqrt{2\alpha-1}$，$\mu=\alpha^{\lambda}$，且 $c=4\alpha^{\alpha}\mathrm{e}^{-\alpha}/[\lambda\Gamma(\alpha)]$。然后，定义 $t(x)=cr(x)$，其中，

$$r(x)=\begin{cases}\dfrac{\lambda\mu x^{\lambda-1}}{(\mu+x^{\lambda})^2}, & x>0\\ 0, & \text{其他}\end{cases}$$

与 $r(x)$相应的分布函数为：

$$R(x)=\begin{cases}\dfrac{x^{\lambda}}{\mu+x^{\lambda}}, & x>0\\ 0, & \text{其他}\end{cases}$$

易于对其进行反变换，得到：

$$R^{-1}(u)=\left(\frac{\mu u}{1-u}\right)^{1/\lambda},\quad 0<u<1$$

为验证 $t(x)$的确是 $f(x)$的强函数，请参见 Cheng(1977)。注意，这是首先指定一个已知

分布来获得强函数的例子($R(x)$实际上是一个对数逻辑斯蒂分布函数(参见第 8.3.11 节)，其形状参数为λ、比例参数为$\mu^{1/\lambda}$、位置参数为 0)，之后再改变密度函数$r(x)$的比例来获得强函数。因此，我们使用反变换法来产生具有密度r的Y。在增加选取的有优势的预检查并改进计算效率后，Cheng(1977)推荐了下述生成算法(预定义常数是$a=1/\sqrt{2\alpha-1}$，$b=\alpha-\ln 4$，$q=\alpha+1/a$，$\theta=4.5$ 和 $d=1+\ln\theta$)：

(1) 产生U_1与U_2为 IID $U(0, 1)$；

(2) 令$V=a\ln[U_1/(1-U_1)]$，$Y=\alpha e^V$，$Z=U_1^2U_2$，且$W=b+qV-Y$；

(3) 若$W+d-\theta Z\geqslant 0$，则返回$X=Y$，否则进入步骤(4)；

(4) 若$W\geqslant\ln Z$，则返回$X=Y$，否则返回步骤(1)。

步骤(3)是增加的预检查，它(如果通过)避免了步骤(4)常规的舍选法测试中的对数计算(如果移去步骤(3)，算法依旧有效，且正是字义上的舍选法)。

如上所述，有几种其他的优秀算法可在$\alpha>1$时使用。Schmeiser 和 Lal(1980)提出了另一种舍选法，其$t(x)$在$f(x)$的“主干”为分段线性化的而在其尾部是指数分布的)；对于α从 1.000 1 到 1 000，这种算法的速度比我们上文提到的方法粗略地说快 1 倍。但是算法更为复杂，并且需要附加时间以对于给定的α值设立必要的常数。这是分析者在选择各种变量产生算法时必须考虑的一种典型的折中。

最后，我们考虑直接使用反变换法来产生伽马分布的随机变量。由于无论伽马分布函数还是其反函数都没有闭合形式，因此我们必须求助于数值方法。Best 和 Roberts(1975)给出了一个数值计算程序对自由度不必是整数的χ^2随机变量的分布函数进行反变换，从而可用于任何$\alpha>0$的伽马随机变量的生成(若$Y\sim\chi_v^2$，其中，$v>0$且不必为整数，则$Y\sim\Gamma(v/2, 2)$。若我们希望得到$X\sim\Gamma(\alpha, 1)$，则首先产生$Y\sim\chi_{2\alpha}^2$，然后返回$X=Y/2$)。IMSL 子例程(Visual Numerics 公司(2004))可用于对χ^2分布函数进行反变换。Press 等(1992，参见第 6.2 节)提供了 C 语言代码以计算χ^2分布函数(对所谓不完整伽马分布函数重新参数化)，然后必须通过求根算法对其进行数值反变换，他们在其书的第 9 章对此进行了讨论。

8.3.5　韦布尔分布

韦布尔分布函数易于反变换得到：

$$F^{-1}(u)=\beta[-\ln(1-u)]^{1/\alpha}$$

由此可得其反变换算法如下：

(1) 产生$U\sim U(0, 1)$；

(2) 返回$X=\beta(-\ln U)^{1/\alpha}$。

我们再次用到U和$1-U$均有相同的$U(0, 1)$分布这一事实，因此，若要用字义上的反变换法，在步骤(2)应用$1-U$替代U。注意到如果Y是具有均值β^α的指数分布，则$Y^{1/\alpha}\sim W(\alpha, \beta)$，本算法依然是正确的，参见第 6.2.2 小节。

8.3.6　正态分布

首先注意到，对于给定的$X\sim N(0, 1)$，通过置$X'=\mu+\sigma X$，我们可得到$X'\sim N(\mu, \sigma^2)$。由此，我们可将注意力限定在产生标准正态随机变量。效率是重要的，因为正态密度常常用于由其他分布舍选法产生随机变量的强函数，例如 Ahrens 和 Dieter(1974)的伽马和β发生器。正态随机变量也可以直接转换成其他分布的随机变量，例如对数正态分布。而且，统计学家在蒙特卡罗研究中试验性地寻找估计值时，对正态性的检验统计的原假设将需要一个有效的正态随机变量的发生源[例子请参见 Filliben(1975)，Lilliefors(1976)，Shapiro 和 Wilk(1965)]。

早期产生$N(0, 1)$随机变量的方法之一是由 Box 和 Muller(1958)提出的，显然仍然

得到应用，尽管有快得多算法可用，但该方法的优点是，保持了所使用的随机数和所产生的 $N(0, 1)$随机变量之间的一一对应关系；因此它可用于在使用公共随机数或者对偶变量法作为方差缩减技术中保持同步的证明(参见第 11.2 和第 11.3 节)。这个方法可以简述为：产生 U_1 与 U_2 为 IID $U(0, 1)$，然后置 $X_1=\sqrt{-2\ln U_1}\cos(2\pi U_2)$ 与 $X_2=\sqrt{-2\ln U_1}\sin(2\pi U_2)$，那么 X_1 与 X_2 是 IID $N(0, 1)$随机变量。由于我们按对得到所要求的随机变量，因此我们可在奇数次调用子程序以刚才描述的方式实际计算 X_1 与 X_2，但只返回 X_1，同时保存 X_2 用于在下一次(偶数)调用时立即返回。这样，我们使用了两个随机数以产生两个 $N(0, 1)$随机变量。虽然如果 U_1 与 U_2 真是 IID $U(0, 1)$，这个方法在原理上是正确的，但若 U_1 与 U_2 实际上是使用线性同余发生器(LCG)生成的相邻随机数(参见第 7.2 节)，实际中的情况就是这样，这就有一个很大的困难。根据第 7.2 节中式(7.1)的迭代，由于 U_2 依赖于 U_1，可以证明，所产生成的随机变量 X_1 与 X_2 必然落在(X_1, X_2)空间的螺线上，而非真正独立正态分布的；其例子请参考文献 Bratley、Fox 和 Schrage (1987，第 223-224 页)。因此，Box-Muller 方法的使用不应采用单流的线性同余发生器，也许有可能代之以使用不同的流或者组合发生器，例如，在附录 7B 中的组合多重递归发生器，或者代之以使用下面介绍的方法之一来产生正态随机变量。

Marsaglia 和 Bray(1964)介绍了一种对 Box-Muller 方法的改进，此算法去掉了三角计算，变成为所谓极坐标法。此方法依赖正态分布本身的特殊性质。Atkinson 和 Pearce (1976)发现此方法，根据所用机器不同，在采用 Fortran 编程计算时比 Box-Muller 方法快 9%～31%(Ahrens 和 Dieter(1972)的实验结果为减少了 27%)。极坐标方法也成对产生 $N(0, 1)$的随机变量，该方法如下：

(1) 产生 U_1 与 U_2 为 IID $U(0, 1)$；对 $i=1, 2$ 令 $V_i=2U_i-1$；且令 $W=V_1^2+V_2^2$。

(2) 若 $W>1$，则返回步骤(1)；否则令 $Y=\sqrt{(-2\ln W)/W}$，$X_1=V_1Y$，$X_2=V_2Y$，则 X_1 与 X_2 是 IID $N(0, 1)$随机变量。

由于步骤(2)可能会发生舍弃 U_1 与 U_2(概率为 $1-\pi/4$，根据习题 8.12)，极坐标法为产生一对 $N(0, 1)$随机变量需要 $U(0, 1)$随机变量个数是随机的。最近，Kinderman 和 Ramage(1976)提出了一种非常快速的产生 $N(0, 1)$随机变量的算法，这种方法更复杂，但是在 FORTRAN 实验中它需要的计算时间比极坐标法的少 30%。

对于要直接使用反变换法生成正态随机变量，人们必须使用数值方法，这是由于正态分布函数及其反函数都没有简单的闭合表达式。Moro(1995)给出了一种这样的方法。IMSL 算法库(Visual Numerics 公司，2004)中也有计算标准正态分布函数反函数的子例程。

8.3.7 对数正态分布

对数正态分布的特性是如果 $Y\sim N(\mu, \sigma^2)$，则 $e^Y\sim LN(\mu, \sigma^2)$。该特性可用于如下算法：

(1) 产生 $Y\sim N(\mu, \sigma^2)$；

(2) 返回 $X=e^Y$。

为完成步骤(1)，可采用在第 8.3.6 小节中所讨论的产生正态随机变量的任何方法。

请注意，此时 μ 与 σ^2 并不是 $LN(\mu, \sigma^2)$分布的均值和方差。实际上，如果 $X\sim LN(\mu, \sigma^2)$，且我们令 $\mu'=E(X)$，$\sigma'^2=\text{var}(X)$，则可得 $\mu'=e^{\mu+\sigma^2/2}$ 和 $\sigma'^2=e^{2\mu+\sigma^2}(e^{\sigma^2}-1)$。因此，如果我们想产生具有给定均值 $\mu'=E(X)$和给定方差 $\sigma'^2=\text{var}(X)$的对数正态分布随机变量 X，我们应在产生中间正态随机变量 Y 前，首先用 μ' 和 σ'^2 解出 μ 与 σ^2。其公式易于得到为：

$$\mu=E(Y)=\ln\frac{\mu'^2}{\sqrt{\mu'^2+\sigma'^2}}$$

$$\sigma^2 = \text{var}(Y) = \ln\left(1 + \frac{\sigma'^2}{\mu'^2}\right)$$

8.3.8　β 分布

首先，注意到我们在区间[0，1]上通过置 $X' = a + (b-a)X$，其中 $X \sim \beta(\alpha_1, \alpha_2)$，就可得到在区间[$a$，$b$]($a<b$)上的 $X' \sim \beta(\alpha_1, \alpha_2)$，因此只考虑 X 在[0，1]区间上的情况就足够了，这样我们称它为 $\beta(\alpha_1, \alpha_2)$分布。

对某些(α_1，α_2)组合而言，$\beta(\alpha_1, \alpha_2)$的某些性质形成了产生贝塔分布随机变量的方法。首先，若 $X \sim \beta(\alpha_1, \alpha_2)$，则 $1-X \sim \beta(\alpha_2, \alpha_1)$，从而，如果我们能易于获得 $\beta(\alpha_1, \alpha_2)$随机变量，我们能容易地产生 $\beta(\alpha_2, \alpha_1)$随机变量。无论 α_1 还是 α_2 为 1 时，都有这种情况发生。例如，若 $\alpha_2 = 1$，则对于 $0 \leqslant x \leqslant 1$，我们有 $f(x) = \alpha_1 x^{\alpha_1 - 1}$，从而分布函数是 $F(x) = x^{\alpha_1}$，且我们可以易于用反变换法，即对于 $U \sim U(0, 1)$，返回 $X = U^{1/\alpha_1}$ 来产生 $X \sim \beta(\alpha_1, 1)$，最后，$\beta(1, 1)$分布就是 $U(0, 1)$。

对于任意 $\alpha_1 > 0$ 与 $\alpha_2 > 0$，产生 $\beta(\alpha_1, \alpha_2)$随机变量的通用方法来自于如下事实：若$Y_1 \sim \Gamma(\alpha_1, 1)$，$Y_2 \sim \Gamma(\alpha_2, 1)$，且 Y_1 与 Y_2 是独立的，则 $Y_1/(Y_1+Y_2) \sim \beta(\alpha_1, \alpha_2)$。这就得到如下算法：

(1) 产生 $Y_1 \sim \Gamma(\alpha_1, 1)$并独立于 Y_1 产生 $Y_2 \sim \Gamma(\alpha_2, 1)$；

(2) 返回 $X = Y_1/(Y_1+Y_2)$。

通过任意合适的产生伽马的方法(参见第 8.3.4 小节)就能产生两个伽马随机变量 Y_1 与 Y_2，因此我们必须注意检查 α_1 与 α_2 是小于 1 还是大于 1。

该方法是相当方便的，倘若我们有对于所有 $\alpha>0$ 的 $\Gamma(\alpha, 1)$发生器，本质上它就能实现；当然，它的效率取决于所选伽马发生器的速度。但是，有相当快(通常也会更加复杂)直接由 β 分布来产生的算法。对于 $\alpha_1 > 1$ 与 $\alpha_2 > 1$，Schmeiser 和 Babu(1980)提出了一种非常快的舍选法，其中的强函数在 $f(x)$中心部分是分段线性的，而在尾部是指数型；快速选取预测试用一个分段线性函数 $b(x)$来定义，$b(x)$小于 $f(x)$(即总在下面)。如果$\alpha_1 < 1$或 $\alpha_2 < 1$(或者两者均是小于 1)，Atkinson 和 Whittaker(1976，1979)，Cheng 和 Jöhnk(1964)都提出了直接产生 $\beta(\alpha_1, \alpha_2)$随机变量的算法。Cheng(1978)的算法 BA 相当简单且同时对于任意 $\alpha_1 > 0$，$\alpha_2 > 0$ 组合都有效；Atkinson(1979a)与 Jöhnk(1964)的算法同样如此。

产生 β 随机变量的反变换法必须依靠数值方法，就像伽马与正态分布一样。Cran、Martin 和 Thomas(1977)用 FORTRAN 程序给出了一个这样的方法，且 IMSL 算法库(Visual Numerics 公司，2013)也有可用的子例程。Press 等(2007，第 6.4 节)给出了 C 代码以计算 β 分布函数(也称为不完整 β 函数)，而后必须通过求根算法对其进行反变换，他们在其书的第 9 章中对此进行了讨论。

8.3.9　皮尔逊 V 型分布

正如在第 6.2.2 小节所说明的，当且仅当 $1/X \sim \Gamma(\alpha, 1/\beta)$时，$X \sim \text{PT5}(\alpha, \beta)$，这就得到下列特性算法：

(1) 产生 $Y \sim \Gamma(\alpha, 1/\beta)$；

(2) 返回 $X = 1/Y$。

第 8.3.4 小节中产生伽马分布的任何算法均可使用，但须注意是 $\alpha<1$，$\alpha=1\alpha<1$，还是 $\alpha>1$。为使用反变换法，我们从第 6.2.2 小节注意到，对于 $x>0$，PT5(α，β)的分布函数为$F(x) = 1 - F_G(1/x)$，其中，F_G是 $\Gamma(\alpha, 1/\beta)$的分布函数。置 $F(X) = U$，则得到 $X = \dfrac{1}{F_G^{-1}(1-U)}$为字义上的反变换法；或者若我们想要利用 $1-U$ 和 U 有同样的 $U(0, 1)$分布这一事实，则得到 $X = 1/F_G^{-1}(U)$。无论在上述哪种情况，我们一般都必须使用数值方法

来计算 F_G^{-1}，正如第 8.3.4 小节中讨论过那样。

8.3.10 皮尔逊 VI 型分布

从第 6.2.2 小节我们注意到，若 $Y_1 \sim \Gamma(\alpha_1, \beta)$ 与 $Y_2 \sim \Gamma(\alpha_2, 1)$，且 Y_1 与 Y_2 是独立的，则 $Y_1/Y_2 \sim \mathrm{PT6}(\alpha_1, \alpha_2, \beta)$，这就直接得到：

(1) 产生 $Y_1 \sim \Gamma(\alpha_1, \beta)$ 并与 Y_1 独立地产生 $Y_2 \sim \Gamma(\alpha_2, 1)$；

(2) 返回 $X = Y_1/Y_2$。

在第 8.3.4 小节中讨论的产生伽马分布任何算法均可使用，但要检查是 $\alpha<1$，或 $\alpha=1$，还是 $\alpha>1$。为使用反变换法，从第 6.2.2 小节注意到，对于 $x>0$，$\mathrm{PT6}(\alpha_1, \alpha_2, \beta)$ 的分布函数为 $F(x)=F_B(x/(x+\beta))$，其中，F_B 是 $\beta(\alpha_1, \alpha_2)$ 的分布函数。置 $F(X)=U$，则得到 $X=\beta F_B^{-1}(U)/[1-F_B^{-1}(U)]$，其中，$F_B^{-1}(U)$ 一般必须用数值方法来计算，正如在第 8.3.8 小节中讨论过的那样。

8.3.11 对数逻辑斯蒂分布

可对数逻辑斯蒂分布函数进行反变换以得到：

$$F^{-1}(u) = \beta\left(\frac{u}{1-u}\right)^{1/\alpha}$$

这就得到反变换算法：

(1) 产生 $U \sim U(0, 1)$；

(2) 返回 $X=\beta[U/(1-U)]^{1/\alpha}$。

8.3.12 有界约翰逊分布

当且仅当 $Z=\alpha_1+\alpha_2\ln[(X-a)/(b-X)] \sim N(0, 1)$ 时 $X \sim \mathrm{JSB}(\alpha_1, \alpha_2, a, b)$，因此我们可以用 Z 来求解该式的 X，得到下述特性算法：

(1) 产生 $Z \sim N(0, 1)$；

(2) 令 $Y=\exp[(Z-\alpha_1)/\alpha_2]$；

(3) 返回 $X=(a+bY)/(Y+1)$。

第 8.3.6 小节产生标准正态分布的任何方法都可以用于步骤(1)中产生 Z。

8.3.13 无界约翰逊分布

$X \sim \mathrm{JSU}(\alpha_1, \alpha_2, \gamma, \beta)$，当且仅当

$$Z = \alpha_1 + \alpha_2 \ln\left[\frac{X-\gamma}{\beta} + \sqrt{\left(\frac{X-\gamma}{\beta}\right)+1}\right] \sim N(0,1)$$

并且我们可以用 Z 来求解该式的 X，得到下述特性算法：

(1) 产生 $Z \sim N(0, 1)$；

(2) 令 $Y=\exp[(Z-\alpha_1)/\alpha_2]$；

(3) 返回 $X=\gamma+(\beta/2)(Y-1/Y)$。

第 8.3.6 小节中的产生标准正态分布任何算法都可用于步骤(1)中产生 Z。该算法的另一种表述为 $X=\gamma+\beta\sinh((Z-\alpha_1)/\alpha_2)$，其中，$Z$ 是步骤(1)中产生的。

8.3.14 贝塞尔分布

正如在第 6.9 节中讨论过的那样，服从拟合的贝塞尔分布的随机变量可以使用 Wagner 和 Wilson(1996b)提出的数值反变换法来产生，该方法需要一个求根算法作为其运算的一部分。

8.3.15 三角分布

首先注意到，如果我们有 $X \sim \mathrm{triang}[0, 1, (m-a)/(b-a)]$，则 $X'=a+(b-a)X \sim$

triang(a，b，m)，所以我们可将注意力限制在 triang(0，1，m)随机变量，其中，$0<m<1$(对于限制 $m=0$ 或 $m=1$ 的情形，则出现左三角或右三角分布，参见习题 8.7)。对于 $0\leqslant u\leqslant 1$,该分布函数易于求逆以得到

$$F^{-1}(u)=\begin{cases}\sqrt{mu}, & 0\leqslant u\leqslant m\\ 1-\sqrt{(1-m)(1-u)}, & m<u\leqslant 1\end{cases}$$

因此，我们说如下反变换法可以用于产生 $X\sim$triang(0，1，m)：

(1) 产生 $U\sim U(0,1)$；

(2) 若 $U\leqslant m$，则返回 $X=\sqrt{mU}$；否则返回 $X=1-\sqrt{(1-m)(1-U)}$。

注意，在步骤(2)中若 $U>m$[注：原文有误]，我们不能使用 U 来代替 X 的公式中的 $1-U$，为什么？另一种产生三角随机变量的方法(使用组合法)请参见习题 8.13。

8.3.16 经验分布

在本节中，我们给出产生服从在第 6.2.4 小节中所定义的经验分布函数 F 和 G 的随机变量的算法。在这两种情形下，都可使用反变换法。

首先假设我们拥有原始的个体观察数据，我们用它们来定义在第 6.2.4 小节中给出的经验分布函数 F(也可参见图 6.24)。虽然反变换法起初会表现出包含某类搜索，但实际上 F 的“拐角”只会准确地出现在 0，$1/(n-1)$，$2/(n-1)$，…，$(n-2)/(n-1)$ 和 1 水平处，这就使我们避免了准确的搜索。下面的算法是反变换法，我们将其校验工作留给读者：

(1) 产生 $U\sim U(0,1)$，令 $P=(n-1)U$，且令 $I=\lfloor P\rfloor+1$；

(2) 返回 $X=X_{(I)}+(P-I+1)(X_{(I+1)}-X_{(I)})$。

请注意，$X_{(i)}$ 必须存储，同时将包含 $X_{(I+1)}-X_{(I)}$ 的值存储在一个分开的数组中会在步骤(2)中避免减法。而且，所产生的 X 值必然总在 $X_{(1)}$ 和 $X_{(n)}$ 之间，这个限制对于使用此方法确定一个经验分布可能是一个缺点。没有搜索使得该算法的边际执行时间本质上与 n 的大小无关，虽然大的 n 需要更多的存储，并且需要更多的准备时间来为 X_i 排序。

下面假设我们的数据是分组的，即我们有 k 个相邻区间$[a_0,a_1)$，$[a_1,a_2)$，…，$[a_{k-1},a_k]$，且第 j 个区间包含 n_j 个观察值，满足 $n_1+n_2+\cdots+n_k=n$。在这种情况下，我们在第 6.2.4 小节中定义过一个经验分布函数 $G(X)$(也可参见图 6.25)，则下列反变换算法产生具有该分布的随机变量：

(1) 产生 $U\sim U(0,1)$；

(2) 找到非负整数 $J(0\leqslant J\leqslant k-1)$，满足 $G(a_J)\leqslant U\leqslant G(a_{J+1})$，并返回 $X=a_J+[U-G(a_J)](a_{J+1}-a_J)/[G(a_{J+1})-G(a_J)]$。

请注意，步骤(2)中找到的 J 满足 $G(a_J)<G(a_{J+1})$，因此在 $n_j=0$ 的区间上是无 X 能产生(而且，显然，$a_0\leqslant X\leqslant a_k$)。步骤(2)中确定 J 的做法可以是直接从左到右进行搜索，或者从 $G(a_{j+1})-G(a_j)$最大的 j 值开始搜索，然后是第二大的，等等。为避免整个地搜索(这需要额外存储的开销)，作为一种替代的办法，我们可以在起初定义一个向量(m_1，m_2，…，m_n)，其中，设定 $n_1m_i=0$，$n_2m_i=1$，以此类推，最后一个设定为 $n_km_i=k-1$(如果某个 $n_j=0$，则对 $j-1$ 不设 m_i。例如，若 $k\geqslant 3$，且 $n_1>0$，$n_2=0$，且 $n_3>0$，则设第一个 n_1m_i为 0，下一个 n_3m_i设为 2)。则步骤(2)中可确定 J 值的办法是：置 $L=\lfloor nU\rfloor+1$，并令 $J=m_L$。这是否值得取决于数据的特定性质，以及取决于运算速度的提升相对于额外存储空间与编程付出的努力的重要程度。最后，Chen 和 Asau(1974)给出了在步骤(2)中确定 J 的另一种方法，该方法基于预计算，对于给定的 U，减小搜索范围；它只需要 10 个额外的内存位置(他们的方法是针对离散经验分布函数的，但也可用于连续的情况)。

在第 6.2.4 小节中简要提到的实验/指数分布也都易于求逆，因此可使用反变换法；文献 Bratley，Fox 和 Schrage(1987，第 151 页)给出了一个精确的算法。

8.4 离散随机变量的产生

本节将讨论产生用于仿真研究的各种离散分布的随机变量的特定算法。正如在第 8.3 节中的那样，我们通常对每种分布给出一种实现起来相当简单且有合理的效率的算法。其他算法在参考文献中给出，这些算法可能快一些，通常以更复杂为代价。

正如在第 8.2.1 小节中所介绍的，离散反变换法可用于任意离散分布，无论可能的取值范围是有限的(可数)还是无限的。本节给出的很多方法是离散反变换法，尽管在某些情况下由于特定方法进行所需要的搜索很好地掩盖了这个事实，这些方法往往利用特殊形式的概率质量函数。但是，就像连续随机变量情况中一样，为产生具有给定分布的随机变量，反变换法或许并不是最有效的方法。

这里应该提到另一种通用方法，它可用于产生任何具有有限范围值的离散随机变量。这就是别名法，由 Walker(1977)提出，而由 Kronmal 和 Peterson(1979)改进，它非常通用且效率高，但它却需要一些初始化准备和额外的存储空间。我们将在第 8.4.3 小节中更详细地讨论别名法，但读者应该记住它可适用于具有有限范围值的任何离散分布(例如二项分布)。对于无限范围的情况，别名法可间接地与通用组合法(参见第 8.2.2 小节)结合起来使用，这也在第 8.4.3 小节中讨论。

除了迭代法以外，还有一些其他通用的产生离散随机变量的思想，例子请参见 Shanthikumar(1985)，Peterson 和 Kronmal(1983)。

最后的说明是有关表面缺失通用性的问题，下面只考虑分布的范围为 $S_n=\{0, 1, 2, \cdots, n\}$ 或 $S=\{0, 1, 2, \cdots\}$，我们原来的离散随机变量的定义具有通用的范围为 $T_n=\{x_1, x_2, \cdots, x_n\}$ 或 $T=\{x_1, x_2, \cdots\}$，与其相比，前者会有更多的限制，但是，其通用性实际上并没有缺失。如果我们真要求随机变量具有质量函数 $p(x_i)$ 和通用的范围 T_n(或 T)，我们可以首先产生具有范围为 S_{n-1}(或 S)的随机变量 I，使得对于 $i=1, 2, \cdots, n$(或 $i=1, 2, \cdots$)有 $P(I=i-1)=p(x_i)$，然后返回随机变量 $X=x_{I+1}$，它具有所要求分布(已知 I，x_{I+1} 可以从 x_i 的存储表来确定或者从计算作为 i 的函数的 x_i 的公式来确定)。

8.4.1 伯努利分布

下述算法相当直观且等价于反变换法(如果将 U 和 $1-U$ 的作用倒过来)：

(1) 产生 $U\sim U(0, 1)$；

(2) 若 $U\leqslant p$，则返回 $X=1$，否则返回 $X=0$。

8.4.2 离散均匀分布

同样，下面给出的直接算法是(严格的)反变换法：

(1) 产生 $U\sim U(0, 1)$；

(2) 返回 $X=i+\lfloor(j-i+1)U\rfloor$。

注意，这里并不需搜索，同时常数 $j-i+1$ 当然应提前计算并存储。

8.4.3 任意离散分布

考虑非常一般的情形，其中在非负整数 S 范围上我们有任意概率质量函数 $p(0)$，$p(1)$，$p(2)$，…，且我们要产生具有相应分布的离散随机变量 X。$p(i)$ 可由某种分布形式理论上确定或直接由一个数据集实验地确定。这里，通过对于所有 $i\geqslant n+1$，置 $p(i)=0$，就包括了有限范围 S_n 的情形(注意，该公式包括了每一种特殊的离散分布形式)。

对于无论是有限范围还是无限范围的情况，直接反变换法如下(空和定义为 0)：

(1) 产生 $U\sim U(0, 1)$；

(2) 返回满足下式的非负整数 $X=I$

$$\sum_{j=0}^{I-1} p(j) \leqslant U < \sum_{j=0}^{I} p(j)$$

注意，对于 $p(i)=0$，该算法永远不会返回 $X=i$，这是因为步骤(2)中两个累加之间的严格不等式不可能成立。步骤(2)需要进行搜索，这可能会花费较多时间。作为一种替代方法，我们可以首先将 $p(i)$进行降序排列，使得搜索会在进行较少次数比较后终止，例子参见习题 8.2。

由于现在是一般的情形，我们给出用于所要求的随机变量是有限范围 S_n的三种其他方法。第一种方法假设每一个 $p(i)$都精确等于一个 q 位十进制小数，为说明此方法，我们设 $q=2$，因此 $p(i)$可表示为 $0.01k_i$，其中，$k_i \in \{0, 1, \cdots, 100\}$为某个整数，$i=0, 1, 2, \cdots, n$，且 $\sum_{i=0}^{n} k_i = 100$。我们对向量(m_1，m_2，…，m_{100})进行初始化，通过设置第一个 $k_0 m_j$ 为 0，下一个 $k_1 m_j=1$，以此类推，而最后一个 $k_n m_j=n$(对于某个 i，若 $k_j=0$，则对于 i，无 m_j 设置)。那么，产生所要求的随机变量 X 的算法如下：

(1) 产生 $J \sim DU(1\ 100)$；

(2) 返回 $X=m_J$。

为完成步骤(1)，请参见第 8.4.2 小节。注意，此方法在步骤(2)需要 10^q个的额外存储位置和一个参考数组，但是，这反变换法，因为所提供的 J 是由第 8.4.2 小节的算法产生的。如果需要 3 位或 4 位十进制小数来准确定义 $p(i)$，则步骤(1)中的值 100 应分别用 1 000或 10 000 来替换，且需要的额外存储空间大小将升高一个或两个数量级。对于某个小 q 值，即使 $p(i)$并不能恰好是 q 位十进制小数，分析人员可以将 $p(i)$舍入到最邻近的百分之一或千分之一以满足精度要求；特别是，当 $p(i)$是直接从数据中得到的比例数，从而精确度不会超过 2 位或 3 位十进制小数时，该方法是一种有吸引力的备选方法。但是在对 $p(i)$进行舍入时，重要的是要记住 $p(i)$的和要正好为 1。

上述方法肯定很快，但如果我们需要高精度的概率，则它会需要大的数据表。Marsaglia(1963)提出了另一种基于表的算法，需要较少的存储空间并只略微增加一点时间。例如，考虑分布为：

$$p(0)=0.15, p(1)=0.20, p(2)=0.37, p(3)=0.28$$

则前面的方法会需要一个长度为 100 的向量来存储 15 0，20 1，37 2 和 28 3。换成：定义两个向量——一个用于 10 分位数，一个用于 100 分位数。为填写 10 分位向量，只看概率中的 10 分位，并按 i(i=0，1，2，3)放进多个拷贝，从而我们也许取一个 0，两个 1，三个 2，两个 3，可得

0 1 1 2 2 2 3 3

类似地，百分位的向量是：

0 0 0 0 0 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3

与概率的百分位相对应。因此，总共有 28 个存储位置(而不是原来的直接表法的 100 个存储空间)。为产生一个 X，选择 10 分位向量的概率等于概率 10 分位的数字之和的十分之一，即

$$\frac{1+2+3+2}{10}=0.8$$

然后随机地(等概率)从 10 分位向量的 8 个数值中选择一个作为返回的 X。另一方面我们选择 100 分位向量的概率等于原来概率的 100 分位的数字之和百分之一，即具有概率

$$\frac{5+0+7+8}{100}=0.2$$

然后再随机地从 100 分位向量的 20 个数值中选择一个作为返回的 X 值。易于看到，本方法的正确性，举例来说，

$$
\begin{aligned}
P(X=2) &= P(X=2\mid \text{选择 10 分位向量})P(\text{选择 10 分位向量}) \\
&\quad + P(X=2\mid \text{选择 100 分位向量})P(\text{选择 100 分位向量}) \\
&= \frac{3}{8}\times 0.8+\frac{7}{20}\times 0.2 \\
&= 0.37
\end{aligned}
$$

这就是所需要的。随着概率的十进制位数的增加，Marsaglia 表的存储优势会变得更加明显；在他最初给出的例子中，有 3 位十进制数，那么前面的直接表法需要使用 1 000 个存储位置，而在本例中 Marsaglia 方法的三个向量(十分位、百分位、千分位)只需要 91 个位置。

用于 X 为有限范围 S_n 的第三个有吸引力方法就是前面提到过的别名法。该方法需要我们首先由给定的 $p(i)$ 计算两个数组，每个长度为 $n+1$。第一个数组包含着所谓截断值的 $F_i\in[0,1]$ $(i=0,1,\cdots,n)$，而第二个数组给出别名 $L_i\in S_i$ $(i=0,1,\cdots,n)$；由 $p(i)$ 计算有效截断值和别名的两个算法在附录 8B 中给出(截断值和别名并不是唯一的，而且附录 8B 中的两种算法对同一分布的确会产生不同结果，但两者都将得到一个有效的变数产生算法)。那么别名法如下：

(1) 产生 $I\sim DU(0,n)$ 并独立于 I 产生 $U\sim U(0,1)$；

(2) 若 $U\leqslant F_I$，则返回 $X=I$，否则返回 $X=L_I$。

因此，步骤(2)中包含着一类“舍去”，但是舍去 I 时我们并不需要从头开始而只是返回 I 的别名 L_I，而不是 I 本身。截断值被视为我们返回 I 而不是其别名 L_I 的概率。产生每个 X 只有一次比较，同时如果像在第 8.4.2 小节中那样产生 I，则对于每个 X，我们只需要两个随机数(只用一个随机数完成步骤 1 的方法，请参见习题 8.17)。虽然准备不复杂，存储截断值和别名需要 $2(n+1)$ 个额外的存储位置。Kronmal 和 Peterson(1979)讨论了一种减少一半存储空间的方法(参见习题 8.18)。无论什么情况下，存储空间总是 n 阶，如果 n 非常大，这就是别名法的原理性不足。

例 8.13 考虑一个随机变量，范围为 $S_3=\{0,1,2,3\}$，概率质量函数为 $p(0)=0.1$，$p(1)=0.4$，$p(2)=0.2$，$p(3)=0.3$。使用附录 8B 中的第一种算法得到如表 8-1 所示的准备数据。

表 8-1

i	0	1	2	3
$p(i)$	0.1	0.4	0.2	0.3
F_i	0.4	0.0	0.8	0.0
L_i	1	1	3	3

举例来说，如果算法的步骤(1)产生 $I=2$，则我们保存 $X=I=2$ 的概率 $F_2=0.8$，否则，我们以概率 $1-F_2=0.2$ 返回 $X=L_2=3$。这样，由于 2 不是任何其他值的别名(即没有任何其他 L_i 等于 2)，当且仅当步骤(1)中 $I=2$ 同时步骤(2)中 $U\leqslant 0.8$ 时，算法返回 $X=2$。因此

$$
\begin{aligned}
P(X=2) &= P(I=2 \text{ 且 } U\leqslant 0.8) \\
&= P(I=2)P(U\leqslant 0.8) \\
&= 0.25\times 0.8 \\
&= 0.2
\end{aligned}
$$

正如所要求的那样，它正好等于 $p(2)$(上面的第二个等式成立是由于 I 和 U 是独立产生的)。另一方面，算法可以两种不同(互斥)的方式返回 $X=3$：若 $I=3$，那么由于 $F_3=0$，因此我们将总是返回 $X=L_3=3$；而若 $I=2$，则我们以概率 $1-F_2=0.2$ 返回 $X=L_2=3$。

因此

$$\begin{aligned}P(X=3)&=P(I=3)+P(I=2 \text{ 且 } U>F_2)\\&=0.25+(0.25\times 0.2)\\&=0.3\end{aligned}$$

这就是 $p(3)$。建议读者也验证对于 $i=0$ 和 $i=1$，该算法也是正确的。图 8.14 表示了该方法的基本原理。图 8.14a 显示出(所有)柱的高度都是 $1/(n+1)=0.25$，从而它是算法步骤(1)中产生的 I 的概率质量函数。柱的阴影部分表示该算法变化的概率质量，每一个阴影部分的数是值 L_i，它作为 X 返回。因此，若步骤(1)产生 $I=0$，则 I 变成其别名 $L_0=1$ 的概率是 $1-F_0=0.6$，并作为 X 返回。0 上方的柱中阴影部分的高度为 $0.6\times 0.25=0.15$，或该柱面积的 60%。相类似的，2 上方的高度为 0.25 的柱的一部分 $1-F_2=0.2$((导致阴影区的高度为 $0.2\times 0.25=0.05$)表示所产生的 $I=2$ 变为 $X=L_2=3$ 的可能性的大小。注意，1 和 3 上面的整个柱全部是阴影，这是因为 F_1 和 F_3 均为 0。然而，所标识的值就是它们自己的别名，所以它们并不会真正改变。图 8.14b 显示了阴影面积变为其目标值后所返回的 X 的概率质量函数，可以看到是等于所要求的概率 $p(i)$。

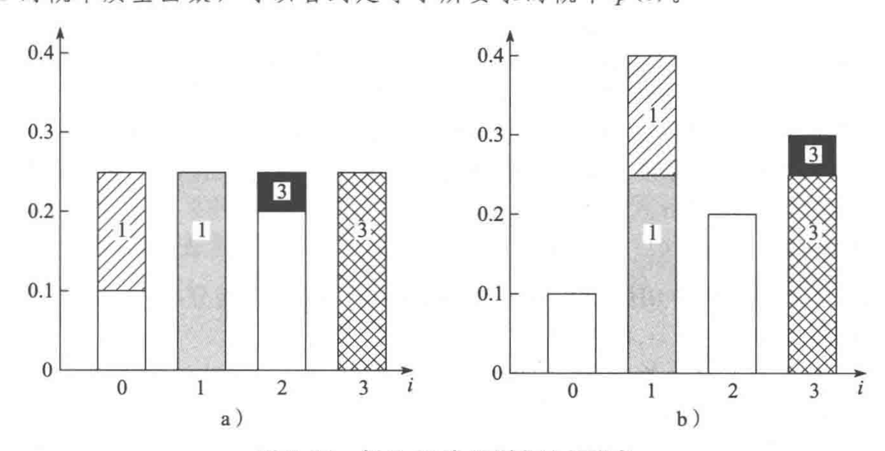


图 8.14 例 8.13 中的别名法的准备 ◀

虽然别名法限于有限范围的离散随机变量，但它与一般的组合法结合起来，可间接地用于无限范围的离散分布，例如几何分布、负二项分布、泊松分布等。举例来说，如果 X 可以是任意非负整数，我们可以检查 $p(i)$，以找到一个 n，使得 $q=\sum_{i=0}^{n}p(i)$ 接近于 1，从而 $X\in S_n$ 的概率非常大。因为对于任意 i，我们可以写出

$$p(i)=q\left[\frac{p(i)}{q}I_{S_n}(i)\right]+(1-q)\left\{\frac{p(i)}{1-q}[1-I_{S_n}(i)]\right\}$$

我们得到下述通用算法：

(1) 产生 $U\sim U(0,1)$，若 $U\leqslant q$，则进入步骤(2)，否则进入步骤(3)；

(2) 使用别名法，在 S_n 上对于 $i=0,1,\cdots,n$，以概率质量函数为 $p(i)/q$，并返回 X；

(3) 使用任何其他方法，在 $\{n+1,n+2,\cdots\}$ 上对于 $i=n+1,n+2,\cdots$，以概率质量函数为 $p(i)/(1-q)$，返回 X。

在步骤(3)中，例如我们可以使用反变换法。由于选择的 n 使得 q 接近于 1，我们可以期望在大多数时间避免进行步骤(3)。

最后，我们要注意，所有上述讨论的基于表的生成方法，以及别名法，都需要初始准备阶段做一些事情。因此，如果随着仿真的推进，概率质量函数随时间变化而频繁变化，则上述各种方法可能就没有吸引力。这种情况下 Rajasekaran 和 Ross(1993)给出了产生一

般离散随机变量的有效方法。

8.4.4 二项分布

为产生 B(t, p)随机变量，请回忆一下第 6.2.3 小节，t 个独立同分布的 Bernoulli(p)随机变量之和具有 B(t, p)分布。该关系得到如下卷积算法：

(1) 产生 Y_1, Y_2, …, Y_t为 IID Bernoulli(p)随机变数；

(2) 返回 $X=Y_1+Y_2+\cdots+Y_t$。

由于此算法的执行时间正比于 t，如果 t 大的话，我们会希望找到另一种方法。一种可能性是具有高效搜索的直接反变换法；另一种是别名法(参见第 8.4.3 小节)，因为 X 的范围是有限的。最后，Ahrens 和 Dieter(1974)，Kachitvichyanukul 和 Schmeiser(1988)都讨论了专门针对二项分布的对于大 t，效率高的算法。

8.4.5 几何分布

下面的算法等价于反变换法，如果我们在步骤(2)用 $1-U$ 代替 U 的话：

(1) 产生 $U \sim U(0, 1)$；

(2) 返回 $X=\lfloor \ln U/\ln(1-p)\rfloor$。

当然，常数 $\ln(1-p)$应预先计算。若 p 接近于 0，则 $\ln(1-p)$也会接近于 0，因此应考虑使用双精度运算以避免步骤(2)中的除法出现过大的舍入误差。对于 p 接近于 1，$\ln(1-p)$将是一个大负数，这也会引起数值运算的困难；幸好，对于大的 p，基于在第 6.2.3 小节介绍过的几何分布和伯努利分布随机变量间的关系，使用一种完全不同算法则效率更高(参详习题 8.14)。

8.4.6 负二项分布

利用第 6.2.3 小节中介绍的负二项分布 negbin(s, p)和几何分布 geom(p)之间的关系得到如下卷积算法：

(1) 产生 Y_1, Y_2, …, Y_s为 IID geom(p)；

(2) 返回 $X=X=Y_1+Y_2+\cdots+Y_s$。

此算法简单，但它的执行时间正比于 s。对于大 s，可以考虑使用 Fishman(1978)讨论过的替代算法，该算法利用了负二项分布、伽马分布和泊松分布之间的特殊关系；它的效率取决于从伽马分布和泊松分布快速产生的能力。其他的备选方法在 Ahrens 和 Dieter (1974)中进行了讨论。

8.4.7 泊松分布

我们产生 $P(\lambda)$随机变数的算法本质上基于第 6.2.3 小节中所说的 $P(\lambda)$与 expo($1/\lambda$)之间的关系。算法如下：

(1) 令 $a=e^{-\lambda}$, $b=1$, $i=0$；

(2) 产生 $U_{i+1} \sim U(0, 1)$，并用 bU_{i+1}代替 b。若 $b<a$，则返回 $X=i$，否则进入步骤(3)；

(3) 使用 $i+1$ 代替 i，并返回步骤(2)。

该算法正确性的证明是，注意到：$X=i$，当且仅当

$$\sum_{j=1}^{i} Y_j \leqslant 1 < \sum_{j=1}^{i+1} Y_j$$

其中，$Y_j=(-1/\lambda)\ln U_j \sim \text{expo}(1/\lambda)$且 Y_j 是独立的。也就是说 $X = \max\left\{i: \sum_{j=1}^{i} Y_j \leqslant 1\right\}$，从而，根据表 6.4 中对泊松分布描述的第一个注解，$X \sim P(\lambda)$。

不幸的是，该算法会随着 λ 的增大而变慢，因为大的 λ 意味着 $a=\mathrm{e}^{-\lambda}$较小，在步骤(2)需要更多的执行时间来进行 U_{i+1} 累积乘使其小于 a(实际上，由于 X 比所需要的 U_{i+1} 的

个数小 1，而步骤(2)的期望执行次数为 $E(X)+1=\lambda+1$，从而执行时间本质上随 λ 增加而线性增长)。一种备选的方法可能是使用别名法加上组合法(由于 X 的范围是无限的)，正如在第 8.4.3 小节所介绍的那样。另一种可能性恐怕是具有高效搜索的反变换法。Atkinson(1979b，1979c)检查过多种这样的搜索程序，并指出，在前面第 8.3.16 小节中讨论过的类似于 Chen 和 Asau(1974)提出的方法的索引搜索执行得较好(这个搜索程序，Atkinson 称作 PQM，需要的准备时间和存储空间数量少但实现起来依然很简单)。其他快速产生泊松随机变量的方法是由 Devroye(1981)，Schmeiser 和 Kachitvichyanukul(1981)给出的。

8.5 随机向量、相关随机变量与随机过程的产生

本章到目前为止，我们实际考虑的只是从各种单变量分布中每次产生单个随机变量。利用独立的随机数组重复应用这些方法的一种由所要求的分布产生一串独立同分布的随机变量。

但是，在某些仿真模型中，我们或许会要从一个指定的联合(或多变量)分布中产生一个随机向量 $\boldsymbol{X}=(X_1, X_2, \cdots, X_d)^{\mathrm{T}}$，其中向量的单个项可能不独立($\boldsymbol{A}^{\mathrm{T}}$表示一个向量或矩阵 $\boldsymbol{A}$ 的转置)。即使我们不能确定 $X_1, X_2, \cdots, X_d$ 准确的全联合分布，我们也可能要产生它们，使得单个 X_i 具有给定的单变量分布(称作 X_i 的边际分布)，并且 X_i 和 X_j 之间的相关系数 ρ_{ij} 由建模型者来规定。在第 6.10 节中我们讨论过对这些情况建模的需求，而在本节中我们将给出产生这种相关随机变量和一些特定情况下的过程的方法的例子。与产生相关随机变量有关的某些其他问题，例如从多变量指数分布中产生，我们不明确地进行讨论。我们推荐读者去参考如下文献：Johnson(1987)、Johnson，Wang 和 Ramberg(1984)、Fishman(1973a，1978)、Mitchell 和 Paulson(1979)、Marshall 和 Olkin(1967)、Devroye(1997)。

8.5.1 利用条件分布

假设我们有给定的全联合分布函数 $F_{X_1,X_2,\cdots,X_d}(x_1, x_2, \cdots, x_d)$，我们希望由它产生一个随机向量 $\boldsymbol{X}=(X_1, X_2, \cdots, X_d)^{\mathrm{T}}$。还假设对于 $i=2, 3, \cdots, d$，已知 $X_j=x_j(j=1, 2, \cdots, i-1)$，我们能得到 X_i 的条件分布；将此条件分布函数记为 $F_i(x_i|x_1, x_2, \cdots, x_{i-1})$(对条件分布函数的讨论请参见文献 Mood，Graybill 和 Boes(1974，第四章)或 Ross(2003，第三章))。另外，令 $F_{X_i}(x_i)$为 $X_i(i=1, 2, \cdots, d)$的边际分布函数。则产生具有联合分布函数 $F_{X_1,X_2,\cdots,X_d}$ 的随机向量 $\boldsymbol{X}$ 的通用算法如下：

(1) 产生具有分布函数 F_{X_1} 的 X_1；

(2) 产生具有分布函数 $F_2(\cdot|X_1)$的 X_2；

(3) 产生具有分布函数 $F_3(\cdot|X_1, X_2)$的 X_3；

⋮

(d)产生具有分布函数 $F_d(\cdot|X_1, X_2, \cdots, X_{d-1})$的 X_d；

($d+1$)返回 $\boldsymbol{X}=(X_1, X_2, \cdots, X_d)^{\mathrm{T}}$。

注意，步骤(2)到步骤(d)中所使用的条件分布函数都用到了前面生成的 X_i；例如：若 x_1 是 X_1 在步骤(1)中生成的值，则步骤(2)中所使用的条件分布函数就是 $F_2(\cdot|x_1)$，以此类推。该算法的有效性证明依赖条件分布的定义并将其留给读者。

该方法会是非常通用的，但其实用性恐怕相当有限。不仅需要定义整个联合分布，还必须进行所需要的边际分布和条件分布的推导。在复杂仿真中恐怕很少能获得这样详细的程度。

8.5.2 多变量正态分布与多变量对数正态分布

具有均值向量 $\boldsymbol{\mu}=(\mu_1, \mu_2, \cdots, \mu_d)^{\mathrm{T}}$、协方差矩阵 $\boldsymbol{\Sigma}$(其中，第(i, j)项为 σ_{ij})的 d 维

多变量正态分布的联合密度函数在第6.10.1小节已经给出。虽然可以应用第8.5.1小节的条件分布函数方法，但Scheuer和Stoller(1962)提出了一种可用的更简单的方法，该方法利用多变量正态分布特性。由于$\boldsymbol{\Sigma}$是对称并且正定的，因此我们可将它唯一地分解为$\boldsymbol{\Sigma}=\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}}$(称为Cholesky分解)，其中$d\times d$矩阵$\boldsymbol{C}$为下三角阵。计算$\boldsymbol{C}$的算法可在Fishman(1973a第217页)、Press等(1992，第2.9节)以及IMSL例程(Visual Numerics公司(2004))中找到。如果c_{ij}是$\boldsymbol{C}$的第(i,j)项元素，则产生所要求的多变量正态向量$\boldsymbol{X}$的算法如下：

(1) 产生Z_1，Z_2，…，Z_d为IID $N(0,1)$随机变量；

(2) 对于$i=1,2,\cdots,d$，令$X_i=\mu_i_\sum_{j=1}^{i}c_{ij}Z_j$，并返回$\boldsymbol{X}=(X_1,X_2,\cdots,X_d)^{\mathrm{T}}$。

为完成步骤(1)中的单变量正态随机变量的产生，请参见第8.3.6小节。用矩阵表示，如果我们令$\boldsymbol{Z}=(Z_1,Z_2,\cdots,Z_d)^{\mathrm{T}}$，则算法就是$\boldsymbol{X}=\boldsymbol{\mu}+\boldsymbol{C}\boldsymbol{Z}$；请注意与已知$X\sim N(0,1)$来产生$X'\sim N(\mu,\sigma^2)$的转换公式$X'=\mu+X$相似。

正如在第6.10.1小节讨论的那样，Jones和Miller(1966)，Johnson和Ramberg(1978)将多变量对数随机向量表示为$\boldsymbol{X}=(\mathrm{e}^{Y_1},\mathrm{e}^{Y_2},\cdots,\mathrm{e}^{Y_d})^{\mathrm{T}}$，其中，$\boldsymbol{Y}=(Y_1,Y_2,\cdots,Y_d)^{\mathrm{T}}\sim N_d(\boldsymbol{\mu},\boldsymbol{\Sigma})$。该关系式定义了向量产生算法：

(1) 产生$\boldsymbol{Y}=(Y_1,Y_2,\cdots,Y_d)^{\mathrm{T}}\sim N_d(\boldsymbol{\mu},\boldsymbol{\Sigma})$；

(2) 返回$\boldsymbol{X}=(\mathrm{e}^{Y_1},\mathrm{e}^{Y_2},\cdots,\mathrm{e}^{Y_d})^{\mathrm{T}}$。

注意，$\boldsymbol{\mu}$和$\boldsymbol{\Sigma}$并不是所要求的多变量对数正态随机向量$\boldsymbol{X}$的均值向量和协方差矩阵，而是相应的多变量正态随机向量$\boldsymbol{Y}$的均值向量和协方差矩阵。X_i的期望值与方差，以及它们之间的协方差和相关系数的公式是在第6.10.1小节中式(6.10)到式(6.13)给出的，并且是多变量正态随机向量$\boldsymbol{Y}$的均值向量$\boldsymbol{\mu}$和协方差矩阵$\boldsymbol{\Sigma}$的函数。因此，随机向量$\boldsymbol{X}$具有已知的均值向量$\boldsymbol{\mu}'=E(\boldsymbol{X})=(\mu_1',\mu_2',\cdots,\mu_d')^{\mathrm{T}}$和具有已知的第$(i,j)$项为$\sigma_{ij}'=\mathrm{cov}(X_i,X_j)$的协方差矩阵$\boldsymbol{\Sigma}'$，如果我们要从$\boldsymbol{X}$产生观测值，则在产生中间的多变量正态随机向量$\boldsymbol{Y}$之前，我们首先应利用$\boldsymbol{\mu}'$和$\boldsymbol{\Sigma}'$求解$\boldsymbol{\mu}$和$\boldsymbol{\Sigma}$。$\boldsymbol{\mu}$的第$i$个元素的公式易于得到，即

$$\mu_i=E(Y_i)=\ln\left(\frac{\mu_i'^2}{\sqrt{\mu_i'^2+\sigma_i'^2}}\right)$$

且$\boldsymbol{\Sigma}$的第(i,j)项是

$$\sigma_{ij}=\mathrm{cov}(Y_i,Y_j)=\ln\left(1+\frac{\sigma_{ij}'}{|\mu_i'\mu_j'|}\right)$$

8.5.3 相关伽马随机变量

我们现在来讨论这样一种情形，这里我们不能写出整个联合分布，而只能确定边际分布(伽马分布)和$\boldsymbol{X}$向量各项间相关系数。的确，有关“多变量伽马”分布应该是什么甚至还没有统一的定义。和多变量正态情形不同，确定了边际分布和相关矩阵这里并不能确定其联合分布。

因此，问题随之产生。对于已知的一组形状参数α_1，α_2，…，α_d，比例参数β_1，β_2，…，β_d，以及相关系数$\rho_{ij}(i=1,2,\cdots,d;j=1,2,\cdots,d)$，我们要产生一个随机向量$\boldsymbol{X}=(X_1,X_2,\cdots,X_d)^{\mathrm{T}}$，使得$X_i\sim\Gamma(\alpha_i,\beta_i)$且$\mathrm{cor}(X_i,X_j)=\rho_{ij}$。直接的困难就是并不是所有的在$-1$和1之间的$\rho_{ij}$值都理论上与给定的一组$\alpha_i$相一致，也就是说，$\alpha_i$对可能的$\rho_{ij}$有限制(参见Schmeiser和Lal(1982))。下一个困难是，即使一组α_i和ρ_{ij}理论上是可能的，但也许没有算法能做这件事。因此，我们只能在某种约束的情况下产生相关的伽马随机变量。

有简单算法的一种情况是双变量情况，$d=2$。进一步的约束是$\rho=\rho_{12}\geqslant0$，即正相关；还有一个约束是$\rho\leqslant\min\{\alpha_1,\alpha_2\}/\sqrt{\alpha_1\alpha_2}$。无论如何，这包含了很多有用的情况，特别是

当 α_1 与 α_2 接近的时候(若 $\alpha_1=\alpha_2$，则去掉 ρ 的上界)。注意到，通过设置 $\alpha_1=\alpha_2=1$，就包括了任何两个正相关的指数随机变量。利用 Arnold(1967)开发的通用技术，算法借助于伽马分布的特性：

(1) 产生 $Y_1 \sim \Gamma(\alpha_1-\rho\sqrt{\alpha_1\alpha_2}, 1)$；

(2) 产生与 Y_1 独立的 $Y_2 \sim \Gamma(\alpha_2-\rho\sqrt{\alpha_1\alpha_2}, 1)$；

(3) 产生与 Y_1 及 Y_2 独立的 $Y_3 \sim \Gamma(\rho\sqrt{\alpha_1\alpha_2}, 1)$；

(4) 返回 $X_1=\beta_1(Y_1+Y_3)$ 和 $X_2=\beta_2(Y_2+Y_3)$。

该方法称为三变量缩减法，这是因为三个随机变量 Y_1、Y_2 与 Y_3 “缩减”为两个最终随机变量 X_1 与 X_2。注意，该算法并不控制 X_1 与 X_2 的联合分布。Schmeiser 和 Lal(1982)指出过这一点。

在某种较少限制情形下也可以产生相关伽马随机变量。Schmeiser 和 Lal(1982)给出了产生具有任意理论上可能的相关性(包括正相关和负相关)的双变量伽马随机向量的算法。Ronning(1977)解决了一般多变量情形(d⩾2)，但也限于考虑一定的正相关。Lewis(1983)给出了一种产生有相同形状参数和比例参数的负相关伽马随机变量的方法。

8.5.4 由多变量族中产生

第 6.10.1 小节中讨论的约翰逊变换系统的多变量版本可以使用 Stanfield(1996)提出的反变换法来产生。所产生的向量将与由样本数据中得到的实验边际矩相匹配，而且只要边际分布歪斜不严重，生成的向量将会有坐标之间的相关性，这些坐标靠近其样本坐标。

正如在第 6.10.1 小节中提到的，贝塞尔分布的产生为双变量的情形。Wagner 和 Wilson(1995)给出了基于第 8.5.1 小节中的条件分布法由相应的随机向量产生样本的方法。Wagner 和 Wilson(1995)指出过将贝塞尔分布扩展到三维或更多维是“可行的但很麻烦的”。有关双变量贝塞尔分布的进一步的结果与方法可在 Wagner 和 Wilson(1996a)中找到。

8.5.5 具有任意规定的边际分布和相关性的随机向量的产生

在第 6.10.2 小节中我们注意到，需要将某些输入随机变量建模成一个随机向量，它具有相当任意的边际分布和相关性结构，而不是将它们作为多变量参数族的成员，如正态分布、对数正态分布、约翰逊分布和贝塞尔分布，来规定和控制其整个联合分布。单个边际分布不必是相同参数族的成员；我们甚至希望规定它们为不同的类型——连续的、离散的或者混合的。唯一的约束就是它们之间相关性结构需要与边际分布的形式和参数保持内在的一致性，正如 Whitt(1976)所讨论的那样，即所规定的相关性结构是可行的。显然，这样的准备工作显然就有了建模的灵活性并易于拟合边际分布，但其后的问题变成如何按照具有这样任意规定的边际分布和相关性的随机变量来产生观测值。本节余下部分，我们将介绍两种处理此问题的方法。

Hill 和 Reilly(1994)介绍了一种方法适用于边际分布要么都是连续或者都是离散的时候，包括具有最可行相关性的分布的组合(或随机混合)，以达到最终的相关性结构是所要求的。他们给出了 $d=3$ 维的特例，此时边际分布为均匀分布、指数分布和离散均匀分布。

Cario 等(2002)开发出了一种称为任意正态变换的方法(NORTA)。应用这一方法，我们将多变量正态随机向量变换为一组随机变量，这些随机变量具有规定的边际分布(包括某些边际分布是连续的，其他是离散的，乃至其他是连续-离散混合的)，以及任意可行的相关性结构。令 F_i 是最终产生的向量 $\boldsymbol{X}=(X_1, X_2, \cdots, X_d)^{\mathrm{T}}$ 的 d 维边际分布函数第 i 个，且令 $\rho_{ij}(\boldsymbol{X})=\mathrm{cor}(X_i, X_j)$是所要求的向量 $\boldsymbol{X}$ 的各项之间的相关性系数。首先，产生标准多变量正态随机向量 $\boldsymbol{Z}=(Z_1, Z_2, \cdots, Z_d)^{\mathrm{T}}$(参见第 8.5.2 小节)，具有 $Z_i \sim N(0, 1)$和相关性系数 $\rho_{ij}(\boldsymbol{Z})=\mathrm{cor}(Z_i, Z_j)$，$\rho_{ij}(\boldsymbol{Z})$的确定下面进行讨论。令 Φ 表示 $N(0, 1)$的分布函数，则产生 $\boldsymbol{X}$ 的第 i 项为 $X_i=F_i^{-1}(\Phi(Z_i))$。由于 $\Phi(Z_i) \sim U(0, 1)$(根据称为概率积分变换的基本结果，例子参见文献 Mood，Graybill 和 Boes(1974，第 202-203 页)的讨

论)，根据产生随机变量的反变换法的有效性，显然 X_i 将有要求的边际分布 F_i。那么，NORTA 方法中的主要任务是要找到 Z_i 间的相关系数 $\rho_{ij}(\boldsymbol{Z})$，它导出所产生的 X_i 间要求的 $\rho_{ij}(\boldsymbol{X})$。Cario 等(2002)给出了一般性地解决这个任务的数值方法，并指出这些方法的效率是相当高的(也可参见 Chen(2001))。还有，$N(0, 1)$分布函数 Φ 会需要数值计算，但有高效率的方法来高精度地做此事，例子，可参见 Abramowitz 和 Stegun(1964)的第 26 章。最后，计算逆分布函数 F_i^{-1} 可能需要数值方法或搜索，这依赖于 F_i 的形式。因此，NORTA 向量产生法需要一些内部的数值计算，虽然在大多数情形下负担应不会太大，特别是与大型、复杂的动态仿真相比来说更是如此。另一方面，NORTA 向量产生法的好处在于单一框架内的通用性和灵活性。

Ghosh 和 Henderson(2002)指出，对于 $d \geqslant 3$，有一组具有可行相关性矩阵的边际分布，NORTA 方法并不能产生。在这些情况下，他们给出了如何修改 NORTA 方法的初始化阶段，使得它会准确匹配边际分布并近似匹配要求的相关性矩阵(也可参见 Ghosh 和 Henderson(2003，2009))。

8.5.6　随机过程的产生

正如在第 6.10.3 小节中已经提到的，有些应用需要产生“相同”随机变量的观测值，就像随时间变化观测它那样。例如，我们想要产生零件在机器上的加工时间序列，或者通信系统中到达信息长度的序列。若假设这个随机变量的观测值是独立同分布的，我们只需要重复地从合适的单变量分布中进行独立采样即可，正如在第 8.1 节到第 8.4 节中所讨论的。

但是，若连续的观测值之间(或相隔多于一次时滞的观测值之间)存在着某种相关性，上述方法未考虑相关性就可能损害模型的有效性，例子参见 Livny(1993)。并行于第 6.10.3 小节中我们讨论的确定或拟合输入随机过程，本节我们将讨论如何能为仿真产生这样的过程。

AR 和 ARMA 过程

对第 6.10.3 小节的式(6.14)中 AR(p)的情形，根据其非常好的定义相当显然地来产生自回归(AR)模型和自回归滑动平均(ARMA)模型。该过程必须以特殊方式进行初始化以得到平稳性，完整详细的内容参见文献 Box 等(2008)。因此，为产生一个 AR(p)，只要机械地简单执行式(6.14)中定义递归，使用第 8.3.6 小节中讲述的任一方法来推进正态分布 ε_i 的产生。在递归过程中通过对参数加权就隐含了所产生过程的自相关结构。X_i 的边际分布为正态分布，这限制了该过程直接用于仿真输入模型。

伽马过程

Lewis 等(1989)讨论了从这类过程的产生，包括指数自回归(EAR)过程。正如第 6.10.3 小节提到的，其结果是一个自相关过程，具有伽马边际分布，并且是一个自回归类型的递归定义和产生过程。

ARTA 过程

任意自回归(ARTA)过程是由 Cario 和 Nelson(1996，1998)开发的，并在第 6.10.3 小节进行了介绍。该过程使建模者对所产生的过程实现任意所要求的平稳边际分布，以及所要求的超过任何规定的滞后 p 的自相关结构。一个指定的 ARTA 过程的产生办法是产生具有 $N(0, 1)$边际分布的 AR(p)基础过程$\{Z_i\}$，之后通过第 6.10.3 小节中的式(6.15)将其变换为所要求的$\{X_i\}$输入过程。正如第 6.10.3 小节中提到的，确定基础$\{Z_i\}$过程的自相关性，从而，用式(6.15)变换后，$\{X_i\}$的自相关结构便是所要求的。为此目的的方法学是在 Cario 和 Nelson(1996)中研究的，而规范说明与产生的软件是在 Cario 和 Nelson(1998)中讨论的。

VARTA 过程

向量任意自回归(VARTA)过程是由 Biller 和 Nelson(2003)开发的，且在第 6.10.3 小

节中进行了介绍。该过程提供了平稳多变量随机过程$\{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots\}$的建模，并产生方法学。

8.6 到达过程的产生

在本节中，我们说明如何产生在第 6.12 小节中讨论的到达过程的到达时间 t_1，t_2，…。

8.6.1 泊松过程

在第 6.12.1 小节中讨论过，具有速率 $\lambda>0$ 的(平稳)泊松过程具有的性质是：到达间隔时间 $A_i=t_i-t_{i-1}$(其中，$i=1, 2, \cdots$)是具有共同均值 $1/\lambda$ 的独立同分布(IID)指数随机变量。因此，我们可以使用如下算法递归地产生 t_i(假设 t_{i-1} 已经确定且我们要产生下一个到达时间 t_i)：

(1) 产生 $U\sim U(0, 1)$独立于任何先前的随机变量；

(2) 返回 $t_i=t_{i-1}-(1/\lambda)\ln U$。

此递归始于计算 t_1(回忆一下 $t_0=0$)。

该算法可以很容易地修改以产生到达间隔时间为 IID 随机变量的任意到达过程，无论是否服从指数分布。步骤(2)中只需要将一个独立的到达间隔时间加到 t_{i-1} 上以便能得到 t_i；上面给出的步骤(2)的形式只是指数到达间隔时间的特殊情形。

8.6.2 非平稳泊松分布

我们现在讨论如何产生服从非平稳泊松过程的到达时间的方法(参见第 6.12.2 小节)。

令人诱惑的办法是修改第 8.6.1 小节中的算法，办法是在步骤(2)用 $\lambda(t_{i-1})$替代 λ 来从已知的 t_{i-1} 产生 t_i。但是，这恐怕是不正确的，正如可以从图 8.15 看出的那样(该图表示 24 小时一天中在某个交叉道口交通到达率)。例如，如果 $t_{i-1}=5$，该错误的“算法”会在 t_i 前产生一个很大的到达间隔时间，因为与 t 在 6 到 9 之间的 $\lambda(t)$相比，$\lambda(5)$小。因此，我们可能会遗漏到达速率即将到来的上升，也就不会产生对应于早高峰的高交通密度；的确，如果生成的 $t_i=11$，则我们将完全遗漏早高峰。Kaminsky 和 Rumpf(1977)使用其他更复杂的近似方法来说明这种危险性。

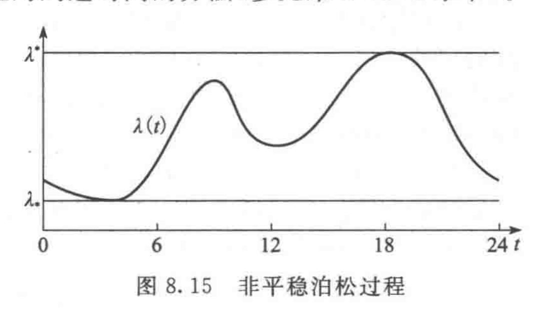


图 8.15　非平稳泊松过程

那么，必须小心才是，要用一种有效方法来产生非平稳泊松过程。可采用 Lewis 和 Shedler(1979)提出的一种通用而简单的方法，称为稀疏法。我们给出稀疏算法的一种特殊情形，它用于 $\lambda^*=\max_t\{\lambda(t)\}$为有限值的时候。简单地说，首先我们产生一个具有恒定速率 λ^* 和到达时间$\{t_i^*\}$的平稳泊松过程(例如，使用第 8.6.1 小节的算法)，然后对 t_i^* 进行“稀疏”，办法是以概率 $1-\lambda(t_i^*)/\lambda^*$ 抛弃(舍去)每一个 t_i^*。这样，如果 $\lambda(t_i^*)$高的话我们将更愿意接受 t_i^* 作为一个到达，从而获得了所要求的性质，即 $\lambda(t)$高的区间到达发生更频繁。一个等效的算法，以一种更方便的递归形式，可表述如下(再一次我们假设 t_{i-1} 已经有效地产生了，我们要产生下一个到达时间 t_i)：

(1) 置 $t=t_{i-1}$；

(2) 产生 U_1 与 U_2 为独立于任何先前随机变量的 IID $U(0, 1)$；

(3) 用 $t-(1/\lambda^*)\ln U_1$ 替代 t；

(4) 若 $U_2\leqslant\lambda(t)/\lambda^*$，返回 $t_i=t$，否则返回步骤(2)。

(再一次说明，此算法也从计算 t_1 开始)。如果计算 $\lambda(t)$较慢(例如，如果 $\lambda(t)$是一个包括指数或三角计算的复杂函数，可能会是这种情况)，则通过增加选取预测试，在步骤

(4)可节省时间；即如果$U_2 \leqslant \lambda_* / \lambda^*$，其中$\lambda_* = \min_t \{\lambda(t)\}$，自动选取当前的$t$值作为下一个到达时间。这一方法在$\lambda(t)$较为平滑时会特别有用。

例 8.14 回忆一下例 6.26，其中，$\lambda(t)$是按图 6.57 所示分段常数函数来实验性地确定的。在图 8.16 中再次绘出该速率函数，连同$\lambda_* = 0.09$与$\lambda^* = 0.84$一起，如所标出的那样。由具有速率λ^*的平稳泊松过程产生$\{t_i^*\}$的值(使用附录 7A 中的随机数发生器的流 3)并在λ^*线上标上叉；正如所期望的，$\{t_i^*\}$的值的确是沿线呈现出是均匀分布的。之后t_i^*按算法所确定的那样被稀疏，并将"选取"的到达在t轴上标上叉。正如所要求的，当$\lambda(t)$低时(例如在时间 11：30 和 11：50 之间)，实际的到达少并且相隔远，因为大多数t_i^*都被稀疏出去了。另一方面，在到达高峰时期(例如从时间 12：00 到 12：20)，大多数t_i^*作为实际的到达都被保留。注意，图 8.16所示的和图 8.9 所示的之间的相似性，这例证了产生随机变量的舍选法是另一种"稀疏"的思路。◀

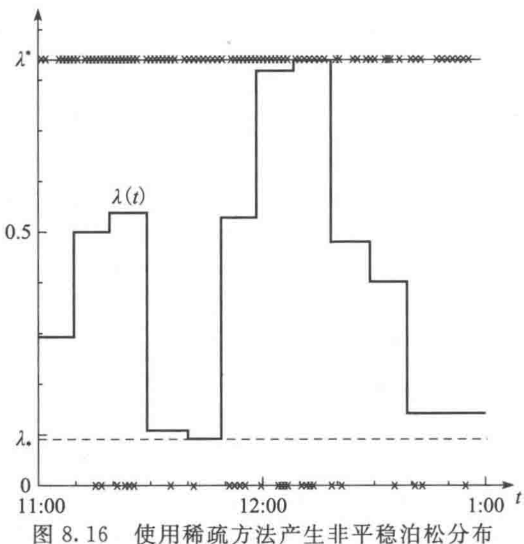


图 8.16　使用稀疏方法产生非平稳泊松分布

虽然细化算法很简单，但在某些情况下它可能效率低。例如，如果$\lambda_{(t)}$除了少数几个高而窄的峰值外，其他相对较小，λ^*在大多数时间比$\lambda(t)$大很多，导致t_i^*被稀疏掉了。在这种情况下，我们可以使用一种具有非常数λ^*曲线的更通用的稀疏方法(参见文献 Lewis 和 Shedler(1979))。

有一种不同的较早的方法，例如，Cinlar(1975，第 94-101 页)提出的，这是产生随机变量的反变换法的模拟方法，正如稀疏方法类似于产生随机变量的舍选法一样。回顾一下第 6.12.2 小节可知，期望函数为：

$$\Lambda(t) = \int_0^t \lambda(y)\mathrm{d}y$$

它总是t的连续函数，因为$\Lambda(t)$是一个不定积分；$\Lambda(t)$是在时间 0 到t之间到达的期望数。因此，通过首先产生一个速率为 1 的泊松到达时间$\{t_i'\}$，之后取$t_i = \Lambda^{-1}\{t_i'\}$，此处$\Lambda^{-1}(t)$为$\Lambda(t)$的反函数，则可产生一个具有期望函数$\Lambda(t)$的非平稳泊松过程。注意，相比于稀疏方法，此方法使用了速率为 1 的所有到达时间$\{t_i'\}$。算法的递归形式如下：

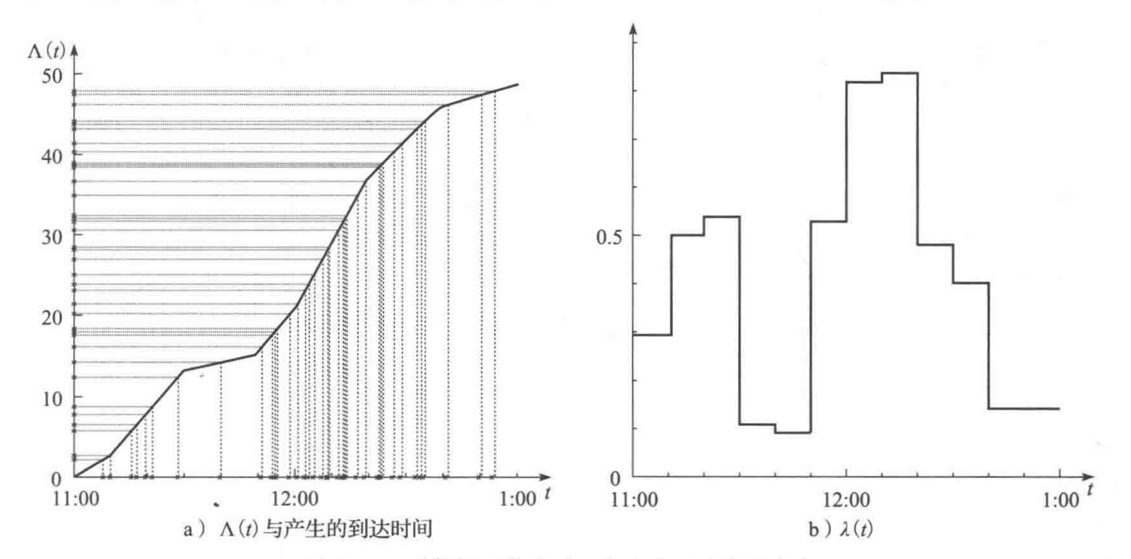
(1) 产生$U \sim U(0, 1)$；

(2) 置$t_i' = t_{i-1}' - \ln U$；

(3) 返回$t_i = \Lambda^{-1}\{t_i'\}$。

例 8.15 图 8.17a 画出了与图 8.16 所示速率函数为$\lambda(t)$对应的期望函数$\Lambda(t)$的图形；为进行对比，我们在图 8.17b 中重画了$\lambda(t)$。注意，$\Lambda(t)$是分段线性的，因为$\lambda(t)$定义为分段常数。而且对于$\lambda(t)$最高的地方的t值，其$\Lambda(t)$上升最陡，即在这个区域到达发生应该最快。稳态速率为 1 的泊松过程的$\{t_i'\}$在$\Lambda(t)$图形的纵轴上标出，可看出，它的分布是相当均匀的。沿着穿过$\Lambda(t)$直角虚线(即取t_i'的$\Lambda^{-1}(t')$)就得到实际到达时间t_i，标在图形的t轴上。显然t_i集中在$\lambda(t)$高的区域(从而$\Lambda(t)$陡，因此"捕获"了很多t_i)，例如在时间 12：00 和 12：20 之间；而在时间 11：30 到 11：50 的低到达阶段，t_i较为分散也很清楚。因此，对纵轴上均匀分布的t_i'运用$\Lambda^{-1}(t')$使其均匀性变形，以满足到达过程的非平稳性。正如反变换法中对均匀分布U运用F^{-1}使U的均匀性变形以满足密度函数f

的思路是一致的。的确，图 8.17所示的和图 8.3所示的之间有很强的相似性。



a）Λ(t)与产生的到达时间　　b）λ(t)

图 8.17 对期望函数求逆以产生非平稳泊松分布

但是，这第二种产生非平稳泊松过程的算法需要 $\Lambda(t)$ 的反函数，这可能有困难(在例 8.15 中，由于 $\Lambda(t)$是分段线性的，通过短搜索和线性插值可能易于实现)。这要与稀疏算法中“浪费”产生的 t_i^* 之间进行权衡，如果速率函数 $\lambda(t)$有一个或多个高而窄的峰值，稀疏法也将变得低效。关于这些方法及其他方法之间的比较的更多内容，请参见 Lewis 和 Shedler(1979)。

最后，我们注意到某些时候有可能利用速率函数 $\lambda(t)$或累积速率函数 $\Lambda(t)$的某些特性以得到一个高效率的算法。这就是在第 6.12.2 小节的末尾引用的某些特殊的估计方法，详细内容请有兴趣的读者参考原始文章。

8.6.3 批到达

考虑这样一种到达过程，其中第 i 批顾客在 t_i时刻到达，同时这批顾客的人数为离散随机变量 B_i。假设离散随机变量 B_i是独立同分布的，同时与 t_i独立。那么，一个产生此到达过程的通用递归算法如下：

(1) 产生下一个到达时间 t_i；

(2) 产生离散随机变量 B_i，其中，B_i与之前的任何 B_j，以及 t_1，t_2，…，t_i都独立；

(3) 返回在 t_i时刻到达 B_i个顾客的信息。

注意，到达时间$\{t_i\}$是任意的；特别是，它们可能来自一个非平稳泊松过程。

附录 8A 舍选法的正确性

我们在这里通过证明，对于任意 x，都有 $P(X \leqslant x) = \int_{-\infty}^{x} f(y)\mathrm{d}y$ 来说明连续随机变量的舍选法是正确的(参见第 8.2.4 小节)。

令 A 表示算法的步骤(3)中选取发生的事件。那么，X 仅定义在事件(或集合)A 上，它是(步骤(1)和步骤(2))定义的 Y 和 U 的整个空间的一个子集。因此，关于 X 的无条件概率声明实际上只是关于 Y 和 U 的条件概率声明(关于 A 的条件)。因为，已知 A 发生，我们有 $X=Y$，我们可写成：

$$P(X \leqslant x) = P(Y \leqslant x \mid A) \tag{8.2}$$

我们将直接计算等式(8.2)的右边。

根据条件概率的定义，即

$$P(Y \leqslant x | A) = \frac{P(A, Y \leqslant x)}{P(A)} \tag{8.3}$$

我们将精确计算出等式(8.3)右边的两个概率值。为此，一个方便的做法是首先注意到，对于任意 y，有

$$P(A | Y = y) = P\left[U \leqslant \frac{f(y)}{t(y)}\right] = \frac{f(y)}{t(y)} \tag{8.4}$$

其中，第一个等式是自然得出的，因为 U 独立于 Y，而第二个等式成立是由于 $U \sim U(0, 1)$ 且 $f(y) \leqslant t(y)$。

我们现在利用等式(8.4)，可以得到：

$$\begin{aligned} P(A, Y \leqslant x) &= \int_{-\infty}^{x} P(A, Y \leqslant x | Y = y) r(y) \mathrm{d}y \\ &= \int_{-\infty}^{x} P(A | Y = y) \frac{t(y)}{c} \mathrm{d}y \\ &= \frac{1}{c} \int_{-\infty}^{x} f(y) \mathrm{d}y \end{aligned} \tag{8.5}$$

下一步，我们注意到，由于 $P(A) = \int_{-\infty}^{+\infty} P(A | Y = y) r(y) \mathrm{d}y = 1/c$（根据式(8.4)以及 f 是密度函数这一事实，从而积分为 1）。该式加上式(8.5)、式(8.3)和式(8.2)，便可得到所要求的结果。

附录 8B 别名法的准备

至少有两种不同的算法用于在第 8.4.3 小节中别名法的准备中计算截断值 F_i 和别名 L_i；对一给定分布这两种方法一般不会产生一组相同的截断值 F_i 和别名 L_i，但两者都是正确的。最初，Walker(1977)用清晰的 FORTRAN 程序给出了下述算法：

(1) 对于 $i=0, 1, \cdots, n$，置 $L_i=i$，$F_i=0$，$b_i=p(i)-1/(n+1)$；

(2) 对于 $i=0, 1, \cdots, n$，进行如下步骤：

a. 令 $c=\min\{b_0, b_1, \cdots, b_n\}$，并令 k 为最小的 b_i 的索引号(可任意地断开关系)；

b. 令 $d=\max\{b_0, b_1, \cdots, b_n\}$，并令 m 为最大的 b_i 的索引号(可任意地断开关系)；

c. 若 $\sum_{j=0}^{n} |b_j| < \varepsilon$，则算法终止；

d. 令 $L_k=m$，$F_k=1+c(n+1)$，$b_k=0$，且 $b_m=c+d$。

注意，如果在某些点满足了步骤(2)c 的条件，则步骤(2)中 i 的剩下的区间将不再执行。步骤(2)c 中的这个条件在理论上其求和式趋向 0，但若坚持为 0 可能会引起浮点计算的困难。在上式中，ε 是一个小的正数，例如 10^{-5}。

虽然上述算法易于用任何编程语言实现，不过 Kronmal 和 Peterson(1979)运用集合运算给出了一种更有效的算法：

(1) 对于 $i=0, 1, \cdots, n$，置 $F_i=(n+1)p(i)$；

(2) 定义集合 $G=\{i: F_i \geqslant 1\}$ 和 $S=\{i: F_i < 1\}$；

(3) 执行下述步骤直到 S 成为空：

a. 从 G 中移出元素 k，从 S 中移出元素 m；

b. 置 $L_m=k$，并使用 F_k-1+F_m 替换 F_k；

c. 若 $F_k<1$，则将 k 放入 S；否则将 k 放入 G。

该算法最少会留下一个 L_i 未定义，但相应的 F_i 值一定等于 1，这样这些别名在变数产生算法中永远不会用到。可以有很多种方法建立在这第二个算法中的集合 S 和 G，例如用一个简单的压入/弹出堆栈或用第 2 章讲述的链表结构。

第二种算法效率更高，这是因为在第一种算法中步骤(2)a 和(2)b 中每个都需要搜索

$n+1$个元素，而在第二种算法中并不需要这种搜索；当 n 很大时这一点变得尤为重要。然而，我们应该注意到，第二种算法中若 $p(i)$之和不严格为 1，则很可能导致数值计算的困难；这是很可能发生的，举例来说，$p(i)$是与数据的频度计数相应的比例数，存在舍入误差。当 $p(i)$之和与 1 相差不过 10^{-5}时，我们经历过第二个算法的失败(使用几个不同的集合实现)。

习题

8.1 给出产生具有下列密度函数的随机变数的算法：

(a) 柯西(Cauchy)分布

$$f(x)=\left\{\pi\beta\left[1+\left(\frac{x-\gamma}{\beta}\right)^2\right]\right\}^{-1},\quad -\infty<\gamma<+\infty,\beta>0,\ -\infty<x<+\infty$$

(b) 冈贝尔(Gumbel)分布

$$f(x)=\frac{1}{\beta}\exp\left(-\mathrm{e}^{-(x-\gamma)/\beta}-\frac{x-\gamma}{\beta}\right),\quad -\infty<\gamma<+\infty,\beta>0,\ -\infty<x<+\infty$$

(c) 逻辑斯蒂(logistic)分布

$$f(x)=\frac{(1/\beta)\mathrm{e}^{-(x-\gamma)/\beta}}{(1+\mathrm{e}^{-(x-\gamma)/\beta})^2},\quad -\infty<\gamma<+\infty,\beta>0,\ -\infty<x<+\infty$$

(d) 帕雷托(Pareto)分布

$$f(x)=\frac{\alpha_2 c^{\alpha_2}}{x^{\alpha_2+1}},c>0,\alpha_2>0,x>c$$

在(a)、(b)与(c)的每一种情形下，对于 $\gamma=0$ 与 $\beta=1$，使用你的算法产生独立同分布的 X_1，X_2，…，$X_{5\,000}$，对于 $n=50$，100，150，…，5 000，给出 $\overline{X}(n)=\sum_{i=1}^{n}X_i/n$ 以验证强大数定律(参见第 4.6 节)，即 $\overline{X}(n)$收敛到 $E(X_i)$(如果存在期望值的话)。对于(d)，以c=1 与 $\alpha_2=2$ 做同样的事情。

8.2 令 X 是离散分布，具有概率质量函数 $p(1)=0.05$，$p(2)=0.05$，$p(3)=0.1$，$p(4)=0.1$，$p(5)=0.6$，以及 $p(6)=0.1$，并对于$i=1$，2，…，6，令 $q(i)=p(1)+p(2)+\cdots+p(i)$。请说服自己，如下方法是用简单地从左到右搜索的离散反变换法：

(1) 产生 $U\sim U(0,1)$，并置 $i=1$；

(2) 若 $U\leqslant q(i)$，则返回 $X=i$，否则进入步骤(3)；

(3) 使用 $i+1$ 替代 i，并返回步骤(2)。

令 N 是步骤(2)执行的次数(从而 N 也是比较的次数)。请证明 N 与 X 是同分布的，从而 $E(N)=E(X)=4.45$。这一算法可使用图 8.18a表示，其中带圆圈的数表示若 U 直接落在带圆圈的数下面的区间中并从左至右搜索所置 X 的值。

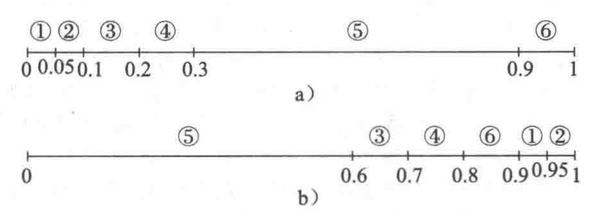


图 8.18 习题 8.2 的两个算法表示

另外，我们可以首先将 $p(i)$进行降序排列并按如下方式组成一个编码向量 $\boldsymbol{i}'(i)$。令 $q'(1)=0.6$，$q'(2)=0.7$，$q'(3)=0.8$，$q'(4)=0.9$，$q'(5)=0.95$，$q'(6)=1$；还令 $i'(1)=1$，$i'(2)=3$，$i'(3)=4$，$i'(4)=6$，$i'(5)=1$，$i'(6)=2$。请证明下述算法是正确的：

(1′) 产生 $U\sim U(0,1)$，并置 $i=1$；

(2′) 若 $U\leqslant q'(i)$，则返回 $X=i'(i)$，否则进入步骤(3)′；

(3′) 使用 $i+1$ 替代 i，并返回步骤(2′)。

如果 N'是本第二个算法的比较次数，请证明：$E(N')=2.05$，不到 $E(N)$的一半。该边际执行时间的节省将取决于特定的分布，相对于编码向量 $\boldsymbol{i}'(i)$的额外的准备时间与存储空间，时间的节省必然更为重要。第二种算法可以使用图 8.18b 表示。

8.3 回忆一下在在 8.2.1 小节中给出的截断分布函数 F^*，以及由它产生随机变量的算法。

(a) 请证明：当 F 是连续且单调增的函数时，第 8.2.1 小节中所说的算法是正确的；

(b) 请证明：下述算法对产生具有分布函数 F^* 的随机变量 X 也是正确的(也假设 F 是连续且单调增的函数)：

(1) 产生 $U\sim U(0,1)$；

(2) 若 $F(a)\leqslant U\leqslant F(b)$，则返回 $X=F^{-1}(U)$，否则返回步骤(1)。

你认为哪一种“更好”？在什么意义上？在什么条件下？

8.4 一个分布函数 F 的截断可以与 8.2.1 小节中的 F^* 不同的形式来定义。仍然是对于 $a<b$，定义分布函数：

$$\widetilde{F}(x)=\begin{cases}0, & x<a\\ F(x), & a\leqslant x<b\\ 1, & b\leqslant x\end{cases}$$

假设我们已经有了产生服从 F 的随机变量的方法，请找到一种产生服从分布函数 $\widetilde{F}$ 的随机变量的方法，并证明你的算法的正确性。

8.5 证明当 F 是单调增函数时第 8.2.1 小节中产生第 i 阶统计量的算法是正确的(提示：利用下面事实，若 U_1，U_2，…，U_n 是 IID $U(0,1)$，则 $U_{(i)}\sim\beta(i,n-i+1)$)。请直接验证，对于 $i=1$ 与 $i=n$，分别令 $V=1-U^{1/n}$ 与 $V=U^{1/n}$ 该方法是正确的。

8.6 推导例 8.3 中双指数分布的反变换法，并将它与该例中组合算法进行比较，你认为哪种算法更好？

8.7 设 $a<b$，右三角分布的密度函数为：

$$f_R(x)=\begin{cases}\dfrac{2(x-a)}{(b-a)^2}, & a\leqslant x\leqslant b\\ 0, & \text{其他}\end{cases}$$

而左三角分布的密度函数为：

$$f_L(x)=\begin{cases}\dfrac{2(b-x)}{(b-a)^2}, & a\leqslant x\leqslant b\\ 0, & \text{其他}\end{cases}$$

上述分布分别记为 $RT(a,b)$ 和 $LT(a,b)$。

(a) 请证明，若 $X\sim RT(0,1)$，则 $X'=a+(b-a)X\sim RT(a,b)$；验证 $LT(0,1)$ 与 $LT(a,b)$ 之间也有相同的关系。这样只需从 $RT(0,1)$ 和 $LT(0,1)$ 中产生随机变量就够了。

(b) 请证明，若 $X\sim RT(0,1)$，则 $1-X\sim LT(0,1)$。这样进一步，将我们的注意力限制在从 $RT(0,1)$ 中产生就够了。

(c) 请推导，从 $RT(0,1)$ 产生随机变量的反变换法；尽管有(b)中的结果，也请推导直接由 $LT(0,1)$ 产生的反变换法。

(d) 作为一种反变换法的替代方法，请证明，若 U_1 和 U_2 是 IID $U(0,1)$ 随机变量，则 $\max\{U_1,U_2\}\sim RT(0,1)$。你是否认为此方法比反变换法好？在什么意义上？(参见例 8.4)

8.8 对于下述每种情况，为产生具有相同分布的随机变量 X，给出一种严格只使用一个随机数方法：

(a) $X=\min\{U_1,U_2\}$，其中 U_1 与 U_2 是 IID$\sim U(0,1)$。

(b) $X=\max\{U_1,U_2\}$，其中 U_1 与 U_2 是 IID$\sim U(0,1)$。

(c) $X=\min\{Y_1,Y_2\}$，其中 Y_1 与 Y_2 是具有共同均值 β 的 IID 指数分布。

将(a)与(b)同习题 8.7 进行比较。将你的(a)到(c)中使用一个 U 的算法与实际产生 U_i 和 Y_i，然后取最大值或最小值的直接算法进行比较。

8.9 8.2.4 小节的通用舍选法有如下的离散模拟形式：令 X 是离散的，对于 $i=0$，±1，±2，…，其概率质量函数为 $p(x_i)$；对于所有 i，令强函数为 $t(x_i)\geqslant p(x_i)$；对于 $i=0$，±1，±2，…，令 $c=\sum_{i=-\infty}^{+\infty}t(x_i)$，且令 $r(x_i)=t(x_i)/c$：

(1) 产生具有概率质量函数 r 的 Y；

(2) 产生与 Y 独立的 $U\sim U(0,1)$；

(3) 若 $U\leqslant p(Y)/t(Y)$，返回 $X=Y$，否则回到步骤(1)再试。

按照类似于附录 8A 的步骤，证明此算法的正确性。在选择 $t(x_i)$ 时哪些因素最重要？

8.10 对于通用的舍选法(无论是第 8.2.4 小节讲述的连续情况，还是习题 8.9 中的离散情况)，求在选取发生前被舍弃的数据对 (Y,U) 数目的分布。请问舍弃的期望数目是多少？

8.11 为从如下密度函数的每一个产生随机变量，给出反变换法、组合法和舍选法。对于每种密度函数，讨论哪种方法更好(首先画出密度函数图)。

(a) $f(x)=\begin{cases}\dfrac{3x^2}{2}, & -1\leqslant x\leqslant 1\\ 0, & \text{其他}\end{cases}$

(b) 对于 $0<a<\frac{1}{2}$，

$$f(x)=\begin{cases}0, & x\leqslant 0\\ \frac{x}{a(1-a)}, & 0\leqslant x\leqslant a\\ \frac{1}{1-a}, & a\leqslant x\leqslant 1-a\\ \frac{1-x}{a(1-a)}, & 1-a\leqslant x\leqslant 1\\ 0, & 1\leqslant x\end{cases}$$

8.12　回忆一下 8.3.6 小节产生 $N(0,1)$随机变量的极坐标法，请证明步骤(2)中“选取”的概率为$\pi/4$；并求 W 在最终“选取”前“舍去”的个数的分布。步骤(1)的执行的期望次数是多少？

8.13　为从第 8.3.15 小节三角分布 triang$(0, 1, m)(0<m<1)$产生随机变量，请给出组合法，并将它与第8.3.15小节中的反变换法作进行比较。(提示：参见习题 8.7)

8.14　(a) 请证明第 8.4.5 小节中给出的从 geom(p)分布产生随机变量的算法的正确性。(提示：对于一个实数 x 和一个整数 i，当且仅当 $i\leqslant x<i+1$ 时，$\lfloor x\rfloor=i$)

(b) 请证明下述算法用于产生 $X\sim$geom(p)的正确性：

(1) 令 $i=0$；

(2) 产生 $U\sim U(0,1)$，并与之前生成的任何随机变数独立；

(3) 若 $U\leqslant p$，则返回 $X=i$；否则用 $i+1$ 替换 i，并返回步骤(2)。

注意，若 p 很大(接近 1)，此算法是一种很有吸引力的替代第 8.4.5 小节给出的算法，这是因为此算法不需要进行对数计算，同时算法会较快终止。

8.15　回忆一下第 6.8 节讨论的偏移的指数分布、伽马分布、韦布尔分布、对数正态分布、皮尔逊 V 型分布、皮尔逊 VI 型分布和对数逻辑斯蒂分布。假设已经能够从以上原有(非偏移)分布产生随机变量，请给出一个由带偏移的上述分布产生随机变量的通用方法(假设偏移参数 γ 已经确定)。

8.16　请给出一个由例 8.7 中的密度函数 $r(x)$产生随机变量 Y 的直接方法，$r(x)$已在图 8.12中给出(也可参见习题 8.7)。

8.17　正如第 8.4.3 小节中所说的，别名法至少需要生成两个 $U(0,1)$随机数——一个用于在步骤(1)中产生 I，另一个用于步骤(2)确定是返回 I 还是其别名。下述别名算法只需要一个随机数，请证明它也是正确的：

(1) 产生 $U\sim U(0,1)$；

(2) 令 $V=(n+1)U$，$I=\lfloor V\rfloor$，且 $U'=V-I$；

(3) 若 $U'\leqslant F_I$，则返回 $X=I$，否则返回$X=L_I$。

(提示：I 与 U'的联合分布是什么。虽然上述“窍门”减少了所产生的随机数的数量，但它恐怕不是一个好主意，这是因为此算法依赖于 $V-I$ 的低阶(至少是有意义的)位是“随机”的，这对很多(伪)随机数发生器来说是有疑问的)

8.18　在附录 8B 给出的别名法的准备算法中，产生的截断值 F_i可能实际上为 1，如果使用第二种准备算法，这将至少会出现在一个 i 上。

(a) 找到一种方法来改变截断值和别名值，使得每一个 F_i都严格小于 1；

(b) 对于所有 F_i 都严格小于 1，找到一种方法，通过将 L_i和 F_i数组组合到一个长度为 $n+1$ 的数组中，使得存储空间需求从 $2(n+1)$降低为 $n+1$。重新说明第 8.4.3 小节的别名算法，使其可以使用一个数组来存储别名和截断值。

8.19　对均匀比法，证明式(8.1)中的 $u(z)$和 $v(z)$公式是正确的。

8.20　证明例 8.8 中接受区域 S 的上下界分别是$v=u$ 和 $v=u^2$。

8.21　开发一个接近例 8.8 中的接受区域 S 形状的强区域 T，s/t 是什么？给出一个在 T 中均匀产生数据点的算法。

8.22　开发用于标准正态分布的均匀比算法，画出其选取域，s/t 是什么？

8.23　标准库马拉斯瓦米(Kumaraswamy)分布具有如下密度函数：

$$f(x)=\alpha_1\alpha_2 x^{\alpha_1-1}(1-x^{\alpha_1})^{\alpha_2-1},\quad 0<x<1$$

其中，$\alpha_1>0$ 和 $\alpha_2>0$ 为形状参数。开发一种基于该分布生成随机变量的算法。

第9章

单系统输出数据分析

9.1 引言

在很多仿真研究中，大量的时间和成本都花费在模型的建立和“编程”上，对于仿真输出数据的合理分析往往非常欠缺。实际上，通常采用的模式是进行一次某种随意时间长度的仿真，然后将所得到的仿真估计作为“真的”模型特性。因为典型的做法是使用来自概率分布的随机采样来驱动仿真模型随时间推进的，这些估计值仅仅只是随机变量的特定实现，而这些取值可能存在很大的方差。因此，通过一次特定的仿真得到的这些估计值可能与模型的真实特性存在很大的差异。当然，其结果是很可能对所研究系统做出错误的推断。

历史上，输出数据分析方法没有以适当的方式进行讨论有几个原因。第一，一些人有令人遗憾的印象，认为仿真最主要的工作就是计算机编程，虽然这确实是一项非常复杂的任务。因此，某些仿真“研究”总是先编制假设文档(参见第5.4.3小节)，接着是“编写程序”，最后通过运行一次仿真以得到“答案”。然而，实际上仿真是一个基于计算机的统计采样试验的过程，所以，如果希望仿真结果有意义，就必须采用合理的统计学方法对仿真试验进行设计和分析。统计分析不恰当的第二个原因是几乎所有仿真的输出过程都是不平稳的，且是自相关的(参见第5.6节)，所以，基于独立同分布观测值的经典统计学方法不能直接应用。直到现在，仍然有几个输出分析问题还没有广泛为人们所接受的解决方法，而且即使现有可用的方法用起来也往往很复杂(例子参见第9.5.4小节)。获得模型的真实参数或者特性的精确估计值的另一个障碍是，收集必要的仿真输出数据需要耗费大量时间。这个困难往往发生在大规模作战问题或者高速通信网络的仿真中。

现在我们将更精确地对仿真输出的随机性进行描述。令 Y_1，Y_2，…为一次仿真运行的输出随机过程(参见第4.3节)。例如，Y_i 可能是某制造系统在第 i 个小时的产量。这时，所有 Y_i 都是随机变量，并且它们往往既不独立，且不是同分布的。因此，第4章中那些基于独立性假设的大多数公式(例如，式(4.12)给出的置信区间)就不能直接使用。

令 y_{11}，y_{12}，…，y_{1m}是使用随机数 u_{11}，u_{12}，…(第 j 次运行时所用的第 i 个随机数记作 u_{ij})对系统进行一次观测长度为 m 的仿真所得到随机变量 Y_1，Y_2，…，Y_m 的一个实现。我们使用另外一组随机数 u_{21}，u_{22}，…运行仿真，那么将得到随机变量 Y_1，Y_2，…，Y_m 的一组不同的实现 y_{21}，y_{22}，…，y_{2m}(这两组实现是不相同的，因为两次仿真运行所使用的随机数不同，它产生输入概率分布的样本也不同)。一般地，假定我们进行重复(运行)n 次长度为 m 的独立仿真(也就是说，在每次重复运行中使用不同的随机数，在每次重复运行的开始统计计数器复位，每次重复运行使用相同的初始状态，参见第9.4.3小节)，得到的观测值为：

$$
\begin{matrix}
y_{11}, & \cdots, & y_{1i}, & \cdots, & y_{1m} \\
y_{21}, & \cdots, & y_{2i}, & \cdots, & y_{2m} \\
\vdots & & \vdots & & \vdots \\
y_{n1}, & \cdots, & y_{ni}, & \cdots, & y_{nm}
\end{matrix}
$$

显然，一次特定的重复运行的观测值不是独立同分布的。然而，请注意，y_{1i}，y_{2i}，…，y_{ni}(第 i 列)是随机变量 Y_i($i=1$，2，…，m)的独立同分布观测值。这种运行之间的独立性

(参见习题 9.1)是本章后面各节将要介绍的相对简单的输出数据分析方法的关键。那么，粗略地说，输出分析的目标就是利用观测值 y_{ji}($i=1, 2, \cdots, m$; $j=1, 2, \cdots, n$)来推断随机变量 Y_1, Y_2, …, Y_m(的分布)。举例来说，$\overline{y}_i(n)=\sum_{j=1}^{n} y_{ji}/n$ 就是 $E(Y_i)$的一个无偏估计。

例 9.1 考虑包含 5 个服务台和一个等待队列的银行系统，每天上午 9 点开门，下午 5 点关门，但要在为下午 5 点之前到来的所有顾客服务完之后结束当天营业。假设顾客的到达服从泊松过程，速率为每分钟到达 1 人(即具有均值为 1 分钟的 IID 指数分布的到达间隔时间)，服务时间为均值为 4 分钟的 IID 指数随机变量，且按先到先服务(FIFO)的方式服务。表 9.1 列出了该银行 10 次独立重复运行仿真得到的几个典型输出统计量。假定开始没有任何顾客。注意，各次重复运行的结果是相当不同的，所以，一次运行得不到"答案"。

表 9.1　银行模型的 10 次独立仿真运行结果

序号	服务人数	完成时间/小时	队列平均延误时间/分钟	队列平均长度	延误时间小于 5 分钟的顾客比例
1	484	8.12	1.53	1.52	0.917
2	475	8.14	1.66	1.62	0.916
3	484	8.19	1.24	1.23	0.952
4	483	8.03	2.34	2.34	0.822
5	455	8.03	2.00	1.89	0.840
6	461	8.32	1.69	1.56	0.866
7	451	8.09	2.69	2.50	0.783
8	486	8.19	2.86	2.83	0.782
9	502	8.15	1.70	1.74	0.873
10	475	8.24	2.60	2.50	0.779

◀

在本章中，我们的目的是讨论仿真输出数据的统计分析方法并提供一些以实际应用为核心的材料，这些材料对于已经具备了概率论和数理统计方面的基本知识的读者是很容易理解的(在阅读本章之前，复习一下第 4 章是可取的)。我们将讨论所有被认为是重要的输出分析方法，但是，重点是那些相对易于理解和使用的统计学方法，这些方法已经在实际中用得很好并能应用于解决实际问题。

在第 9.2 节和第 9.3 节中，我们讨论关于输出分析的仿真的类型，以及每类仿真中性能或参数 θ 的度量方法。第 9.4 节到第 9.6 节将说明如何得到每种参数 θ 的点估计$\hat{\theta}$以及置信区间，在确定置信区间时还需要估计$\hat{\theta}$的方差，即 $\widehat{\text{var}}(\hat{\theta})$。所讨论的每种分析方法会遇到下面的一个或两个问题：

(1) $\hat{\theta}$不是 θ 的无偏估计，即 $E(\hat{\theta})\neq\theta$，见第 9.5.1 小节的例子。

(2) $\widehat{\text{var}}(\hat{\theta})$不是 $\text{var}(\hat{\theta})$的无偏估计，见第 9.5.3 小节的例子。

第 9.7 节将上述分析扩展到同时有若干不同参数的置信区间的构造。最后，在第 9.8 节中，我们介绍如何使用重要变量的时间图深入分析系统的动态性能。

我们并不企图给出输出数据分析所有相关的参考文献，因为这方面的文章数以百计。Law(1983)的综述对截至 1983 年的相关研究进行了总结；Pawlikowski(1990)的综述、Alexopoulos 和 Seila(1998)以及韦尔奇(Welch)(1983)著作中的相关章节也与此有关。许多近期的文章发表在"Transactions on Modeling and Computer Simulation"和"Operations Research"等期刊以及冬季仿真会议(每年 12 月举行)的论文集中。

9.2 随机过程的瞬态和稳态行为特性

考虑输出随机过程 Y_1，Y_2，…。令 $F_i(y|I)=P(Y_i \leqslant y|I)(i=1, 2, \cdots)$，其中，$y$ 为一个实数，I 表示系统在仿真开始的 0 时刻所用的初始条件[条件概率 $P(Y_i \leqslant y|I)$是在给定初始状态 I 的条件下事件$\{Y_i \leqslant y\}$发生的概率]。对于一个制造系统而言，I 可能规定 0 时刻现有的作业数量，以及每台机器是忙或是闲。我们称 $F_i(y|I)$为初始条件 I 下在(离散的)i 时刻输出过程的瞬态分布。注意，通常情况下，对于不同的 i 值和不同的初始条件 I，$F_i(y|I)$也是不同的。对于特定的一组初始条件 I，以及递增的时间索引 i_1，i_2，i_3 和 i_4，图 9.1 给出了对应于随机变量 Y_{i_1}，Y_{i_2}，Y_{i_3}，以及 Y_{i_4} 的瞬态分布的密度函数，其中假设随机变量 Y_{i_j} 的密度函数为 $f_{Y_{i_j}}$。密度函数 $f_{Y_{i_j}}$ 说明了随机变量 Y_{i_j} 从一次重复运行到另一次重复运行是如何变化的。特别地，假设对于仿真进行了很多次重复运行，记为 n，并观测每一个随机过程 Y_1，Y_2，…。如果做出随机变量 Y_{i_j} 的 n 个观测值的直方图，则所得到的直方图(比例合理时)就很接近密度函数 $f_{Y_{i_j}}$。

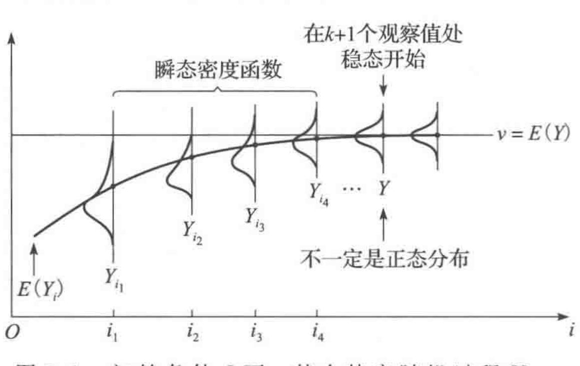


图 9.1 初始条件 I 下，某个特定随机过程 Y_1，Y_2，…的瞬态和稳态密度函数

对于固定的 y 和 I，概率 $F_1(y|I)$，$F_2(y|I)$，…是一组数。对于所有 y 和任何初始条件 I，当 $i \to +\infty$时，如果 $F_i(y|I) \to F(y)$，则称 $F(y)$为输出过程 Y_1，Y_2，…的稳态分布。严格地说，稳态分布 $F(y)$只有在 $i \to +\infty$的极限时才能得到。然而在实际中，往往存在一个有限的时间索引点，比如 $k+1$，使得从此点开始的分布每点将近似相同；像图 9.1 显示的那样，从 $k+1$ 时刻起称“稳态”开始。注意，稳态并不意味着在一次特定的仿真中随机变量 Y_{k+1}，Y_{k+2}，…都取相同的值，而是意味着它们都近似地具有相同的分布。进一步，这些随机变量之间并不独立，但近似组成了一个方差平稳的随机过程(参见第 4.3 节)。韦尔奇(1983)对瞬态和稳态分布进行了极好的讨论。

稳态分布 $F(y)$与初始条件 I 无关，但是，瞬态分布 $F_i(y|I)$到 $F(y)$的收敛速度与初始条件有关，正如下面的例子所表明的那样。

例 9.2 考虑 $\rho=0.9(\lambda=1, \omega=10/9)$的 $M/M/1$ 队列的随机过程 D_1，D_2，…，其中 D_i 表示第 i 个顾客在队列中的延误时间，模型中。在图 9.2 中我们画出了在 0 时刻各种系统中顾客数 s 的情况下，随着 i 增加而逐渐增大，瞬态均值 $E(D_i)$到稳态均值 $d=E(D)=8.1$ 的收敛过程(随机变量 D 具有在队列分布中的稳态延误)。注意，令人惊讶的是，$s=15$ 情况下的 $E(D_i)$收敛到 d 的速度远远快于 $s=0$ 时的情况(参见习题 9.11)。$E(D_i)$的值是由 Kelton 和 Law(1985)推导出来

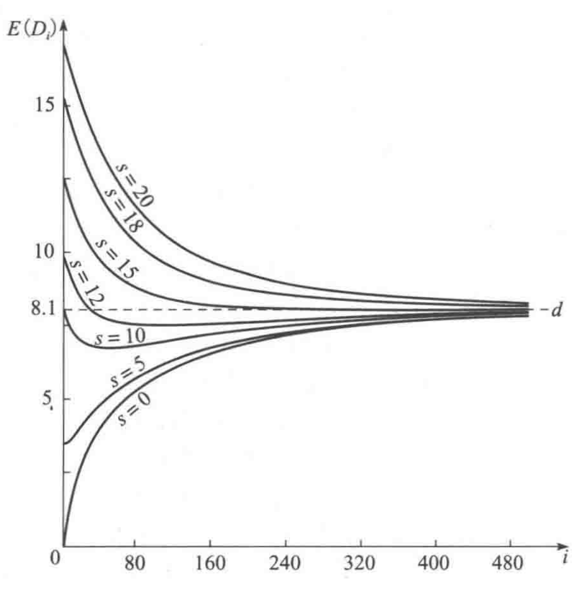


图 9.2 $\rho=0.9$ 的 $M/M/1$ 队列，$E(D_i)$是 i 与 $t=0$ 时刻系统中顾客数量 s 的函数

的，还可以参考文献 Kelton(1985)以及 Murray 和 Kelton(1988)。D 的分布函数由附录4A 中的式(4.14)给出。◀

例 9.3 考虑例 4.23 库存问题中的随机过程 C_1，C_2，…，其中 C_i 表示第 i 个月的总费用。在图 9.3 中，我们绘制了初始库存水平为 57 情况下，随着 i 增加而逐渐增大，$E(C_i)$ 收敛到稳态均值 $c=E(C)=112.11$ 的过程[参见 Wagner(1969，第 A19 页)]。注意，收敛过程显然不是单调的。◀

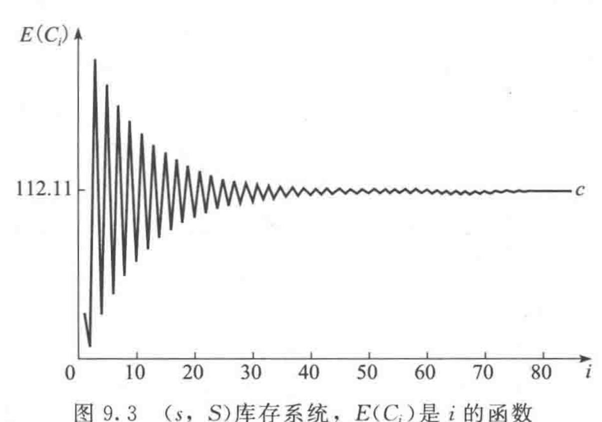


图 9.3 (s，S)库存系统，$E(C_i)$是 i 的函数

在例 9.2 和例 9.3 中我们绘制了期望值 $E(Y_i)$收敛到稳态均值 $E(Y)$的过程。然而，应该记住，随着 i 增大，Y_i 的分布也收敛到 Y 的分布。

9.3 关于输出分析的仿真类型

可用于设计和分析仿真实验的方法取决于所进行仿真的类型，如图 9.4 所描述的那样。根据是否有明确的方法来确定运行长度，仿真可以分为终止型的和非终止型。此外，如图 9.4 所示，非终止型仿真的性能和参数也有多种度量方式。这些概念在下面进行更精确的定义。

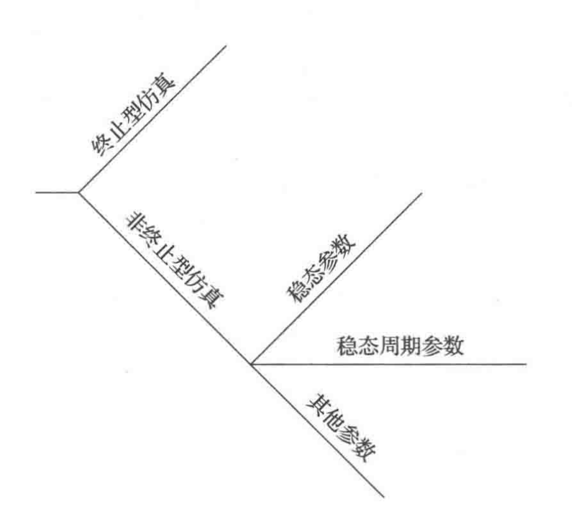


图 9.4 有关输出分析的仿真类型

终止型仿真是指有一个“自然的”事件 E 来规定每次运行(重复运行)的长度的仿真。因为不同的运行使用独立的随机数和相同的初始化规则，这意味着来自不同运行的可比较的随机变量之间是独立同分布的(参见第 9.4 节)。事件 E 通常发生在下面一些时间点，如系统被“清空”时(参见例 9.4)；在某个时间点，超过该点没有获得任何有效信息(参见例 9.5)；或者决策者预先指定的时间点(参见例 9.8)。事件 E 在任何运行之前指定，但是对特定的运行来说 E 发生的时间也许是一个随机变量。因为终止型仿真的初始条件一般影响所要求的性能度量，所以这些初始条件应当代表实际系统的情况(见第 9.4.3 小节)。

例 9.4 一个零售/商业机构，例如银行，每天晚上停止营业。如果该机构每天从上午 9 点到下午 5 点开门，仿真的目标可能是估计上午 9 点开始到下午 5 点关门前进入的最后一个顾客服务结束这整个时间区间的顾客服务质量。这种情况下，E={仿真时间至少过去了 8 个小时且系统为空}，仿真的初始条件是在 0 时刻已有的顾客数为 0(参见第 9.4.3 小节)。◀

例 9.5 考虑一场蓝军和红军的地面军事对抗。仿真的目的也许是根据某种初始兵力强弱来确定战斗结束时(最终)的兵力强弱。这种情况下 E={蓝军或者红军取得战斗的胜利}。战斗结束的条件，举例说，可以是一方兵力损失超过 30%，因为该方恐怕不再有生存的能力。仿真初始条件的选择，例如，每方兵力的部队和坦克数量，这里一般不存在问题，因为是由所考虑的军事场景来确定的。◀

例 9.6 某飞机制造商接到了一个生产 100 架飞机的订单，它必须在 18 个月内交货。公司希望通过各种生产配置进行仿真，看看哪种配置能以最小的成本满足交货时间要求。这种情况下，E={完成 100 架飞机}。◀

例 9.7 考虑某制造公司，每天的工作时间为 16 小时(两个班次)，当天未完成的在制品留到第二天继续加工。这会用 E={仿真时间过去 16 个小时}将仿真限定为终止型仿真吗？答案是否定的，因为该制造运行本质上是一个连续过程，前一天的结束的条件就是后一天的初始条件。◀

例 9.8 某个销售单一产品的公司希望决定长度为 120 个月的计划期内，库存数量应为多少(参见第 1.5 节)。已知某个初始库存水平，目的可能是确定每个月的订单是多少，以使得运行该库存系统每月期望平均成本最小化。这种情况下，E={仿真了 120 个月}，并且用当前的库存水平进行仿真的初始化。◀

非终止型仿真是指无自然事件 E 来规定一次运行长度的仿真。这通常发生在我们正在设计一个新系统或者改变现有系统的时候，而且我们关心的是系统在“正常”运作时的长期运行的行为特性。然而，“长期运行”无法自然地转换为一个终止事件 E。这种仿真的性能度量称为稳态参数，如果它表征了某个输出随机过程 Y_1，Y_2，…稳态分布的话。在图 9.1 中，如果随机变量 Y 有稳态分布，则我们也许会对估计稳态均值 $\nu=E(Y)$或者对某个实数 y 的概率 $P(Y\leqslant y)$感兴趣。

例 9.9 考虑某公司正在建造一个新的制造系统，新制造系统经过足够长的时间运行以后，工人熟悉其任务并且机械加工的困难已经找到，公司希望确定该系统长期运行(稳态)平均每小时的产量。假设：

(a) 系统的工作时间为每天 16 小时，每周 5 天。

(b) 在每一班生产结束和下一班开始时的生产损耗可以忽略(参见习题 9.3)。

(c) 不中断(例如，午休)，中断是指每天在规定的时间停产。

可以采用“笼统的”16 小时日对这个系统进行仿真，从而忽略在每天末尾以及周末的系统空闲时间。令 N_i 是第 i 个小时生产的零件数。如果随机过程 N_1，N_2，…对于随机变量 N 存在稳态分布，则我们感兴趣的是估计均值 $\nu=E(N)$(参见习题 9.4)。◀

应该提及的是，大多数实际系统的随机过程并没有稳态分布，因为系统的特征是随时间变化而变化的。例如，在制造系统中，生产调度规则、设备布局(例如机器的数量和位置等)都有可能随时变化。另一方面，仿真模型(它是实际的一种抽象)可以有稳态分布，因为模型的特征通常假定是不随时间变化而变化的。当有了系统特征的新信息时，可重做稳态分析。

在例 9.9 中，假如制造公司想知道系统从启动到以“正常的”方式运行所需要的时间，这恐怕是终止型仿真，终止事件 E={被仿真的系统正在“正常”运行}(如果能够这样定义的话)。因此，一个特定系统的仿真既有可能是终止型，也有可能是非终止型，这取决于仿真研究的目标。

例 9.10 考虑一个目前尚不存在的通信网络的仿真模型。因为尚无典型的信息到达机制的代表性数据，通常假设信息的到达服从恒定速率的泊松过程，速率等于峰荷时期信息的预期到达速率(当系统实际已经建立起来时，到达速率的变化是时间的函数，峰荷的区间会相对短一些)。由于在“正常运行”下系统的状态是未知的，初始条件的确定必然有某种随意性(例如，在 0 时刻无信息出现)。那么，目标是，仿真运行足够长，使得随意确定的初始条件对系统性能估计度量(例如一条信息的平均端到端延迟)不再有明显的影响。

在对建议的通信网络进行上述稳态分析时，我们本质上是试图确定网络对无限持续时间的峰荷如何做出响应。然而，假如在实际网络中峰值期很短，或者假如峰值期之前的到达速率远远小于峰值速率，我们的分析有可能过高估计网络在峰值期的阻塞程度。这有可能导致所购买的网络配置超出实际的需要。◀

考虑用无稳态分布的随机过程 Y_1，Y_2，…进行非终止型仿真。假定我们将时间轴划分成等长，相邻的时间区间称为周期。(例如，在制造系统中，周期可能为一个 8 小时班)。令 Y_i^C 为定义在第 i 个周期上的随机变量，且假设 Y_1^C，Y_2^C，…是可比的。假定过程 Y_1^C，Y_2^C，…具有稳态分布 F^C，且 $Y^C \sim F^C$。那么称某个性能的度量为稳态周期参数，如果它是 Y^C 的一个特征的话，比如周期 $\nu^C = E(Y^C)$。因此，稳态周期参数只是一个合适的周期过程 Y_1^C，Y_2^C，…的稳态参数。

例 9.11 在例 9.9 的制造系统中，假设每 8 小时轮班中第 5 个小时开始时有半小时的午休时间，则每小时的产量 N_1，N_2，…这个过程具有非稳态分布(参见习题 9.6)。令 N_i^C 为第 i 个 8 小时轮班(周期)中平均每小时产量，那么我们也许关心的是估计一个周期内稳态期望平均小时产量，$\nu^C = E(N^C)$，这就是一个稳态周期参数。◀

例 9.12 考虑某航线的呼叫中心。假设呼叫到达系统的速率在每天的不同时刻和每周中的不同天是不同的，但是周到周的到达速率模式是相同的。令 D_i 是第 i 个到达呼叫经历的延迟时间，则随机过程 D_1，D_2，…没有稳态分布。令 D_i^C 为第 i 周的平均延迟时间，那么，我们可能关心的是，估计一周内的稳态期望平均延迟时间，$\nu^C = E(D^C)$。◀

对于某非终止型仿真，假定随机过程 Y_1，Y_2，…没有稳态分布，并且无合适的周期定义使得相应的随机过程 Y_1^C，Y_2^C，…具有稳态分布。例如，如果模型参数随着时间变化而不断发生变化，则有可能出现这种情况。在例 9.12 中，如果呼叫的到达速率每周、每年都不断地发生变化，就有可能不好定义稳态(周期)参数。然而，在这种情况下，典型的做法是固定描述输入参数如何随时间变化而变化的数据的数量。在效果上，这为仿真提供了一个终止事件 E，从而第 9.4 节中的终止型仿真的分析方法在这里就适用了。这就是为什么我们在本章后面的内容中并没有将这种情形作为单独的情形来讲的原因。这类仿真的性能和参数的度量通常随时间变化而发生变化，因而包含在图 9.4 中的“其他参数”类中。

例 9.13 考虑例 5.26 中的制造系统。根据市场情况，有一份 3 个月的生产计划，其中描述了每一周所生产的计算机种类及数量。考虑到销售变化和新计算机的引入，该计划每周都发生变化。这种情况下，每周和每月的产量都没有稳态分布，从而我们运行一个长度为 3 个月的终止型仿真，并估计每周的平均产量。◀

9.4　终止型仿真的统计分析

假设我们进行 n 次独立重复运行的终止型仿真，其中每次重复运行都以事件 E 结束，且以“相同的”初始条件开始(参见第 9.4.3 小节)。重复运行的独立性是通过对每次重复运行采用不同的随机数流来实现的(关于如果在多于一次执行中完成 n 次重复运行，如何易于实现的讨论，请参见第 7.2 节)。为了简单起见，假设所关心的系统性能度量只有一个(在第 9.7 节中该假设取消)。令 X_j 为定义在第 j 次重复运行时的随机变量($j=1, 2, \cdots, n$)，假设不同次重复运行得到的 X_j 是可比的，则所有 X_j 为独立同分布的随机变量。对例 9.1 和例 9.4 的银行系统来说，X_j 可以是第 j 次重复运行中一天内的平均延迟时间 $\sum_{i=1}^{N} D_i/N$ (见表 9.1 的第 4 列)，其中 N(随机变量)为一天中服务的顾客数。对例 9.5 的战斗模型来说，X_j 可以是在第 j 次重复运行中被摧毁的红军坦克数量。最后，对例 9.8 的库存系统来说，X_j 可能是在第 j 次重复运行中的平均成本 $\sum_{i=1}^{120} C_i/120$ 。

9.4.1　均值估计

假设我们希望得到均值 $\mu = E(X)$ 的点估计和置信区间，其中，X 为如上所述的重复运行中定义的随机变量。独立运行 n 次仿真的重复运行，并令 X_1，X_2，…，X_n 是所得到的独立同分布的随机变量。那么，将所有 X_j 代入式(4.3)和式(4.12)中，可得到 $\overline{X}(n)$ 为

μ 的无偏点估计，而且 μ 的百分之 $100(1-\alpha)(0<\alpha<1)$ 置信区间为：

$$\overline{X}(n) \pm t_{n-1,1-\alpha/2}\sqrt{\frac{S^2(n)}{n}} \tag{9.1}$$

其中，样本方差 $S^2(n)$ 由式(4.4)给出。我们称基于式(9.1)的置信区间为固定样本长度。

例 9.14 对于例 9.1 的银行系统，假设希望得到在一天中一个顾客的期望平均延误时间的点估计和近似 90%的置信区间，每个顾客的期望平均延误时间为：

$$E(X) = E\left[\frac{\sum_{i=1}^{N} D_i}{N}\right]$$

注意，我们估计的是期望平均延误时间，因为通常情况下每个延误时间有不同的均值。根据表 9.1 中 10 次重复运行的结果，我们得到：

$$\overline{X}(10) = 2.03,\quad S^2(10) = 0.31$$

并且

$$\overline{X}(10) \pm t_{9,0.95}\sqrt{\frac{S^2(10)}{10}} = 2.03 \pm 0.32$$

因此，根据对这些置信区间的正确解释(参见第 4.5 节)，我们有近似 90%的把握说 $E(X)$ 落在区间[1.71，2.35]分钟上。◀

例 9.15 对于第 1.5 节和例 9.8 的库存系统，假设我们希望得到 120 个月计划期内的期望平均成本的点估计和近似 90%的置信区间，期望平均成本为：

$$E(X) = E\left[\frac{\sum_{i=1}^{120} C_i}{120}\right]$$

进行 10 次独立的重复运行并得到如下的 X_j：

129.35	127.11	124.03	122.13	120.44
118.39	130.17	129.77	125.52	133.75

从而有

$$\overline{X}(10) = 126.07,\quad S^2(10) = 23.55$$

以及 90%的置信区间为：

$$126.07 \pm 3.47 \text{ 或}[122.60, 129.54]$$

注意，方差系数(参见表 6.5)是一种可变性的度量，对库存系统来说方差系数是 0.04，而对银行模型来说方差系数是 0.27。也就是意味着，银行模型的 X_j 比库存系统的变化大。◀

例 9.16 对于例 9.1 的银行系统，假设我们希望得到一天中延误时间少于 5 分钟的期望的顾客比例的点估计和近似 90%的置信区间，期望的顾客比例为：

$$E(X) = E\left[\frac{\sum_{i=1}^{N} I_i(0,5)}{N}\right]$$

其中，对于 $i=1, 2, \cdots, N$，示性函数 $I_i(0, 5)$ 定义为：

$$I_i(0,5) = \begin{cases} 1, & D_i < 5 \\ 0, & D_i \geq 5 \end{cases}$$

从表 9.1 的最后一列，可得到：

$$\overline{X}(10) = 0.853,\quad S^2(10) = 0.004$$

故 90%的置信区间为：

$$0.853 \pm 0.036 \text{ 或者}[0.817, 0.889]$$ ◀

式(9.1)所给出的置信区间(用接近 $1-\alpha$ 的覆盖度)的正确性依赖于 X_j 是正态随机变

量(或 n “足够大”)的假设，这就是为什么我们称例 9.14、例 9.15、例 9.16 中的置信区间为近似 90%的置信区间。因为这种假设在实际中很少会得到满足，所以现在用几个具有已知均值的简单随机模型试验性地研究置信区间偏离正态性的鲁棒性。目的是为仿真工程技术人员提供某种指南，以便了解在实际中根据覆盖度来判断置信区间的可信度究竟是怎样的。

首先对 $\rho=0.9$ 的 $M/M/1$ 排队系统进行 500 次独立仿真实验。在每次实验中考虑 $n=$ 5、10、20、40，且对于 n 的每个取值，用式(9.1)构建下面随机变量的近似 90%的置信区间：

$$d(25|_{s=0})=E\left(\left.\frac{\sum_{i=1}^{25}D_i}{25}\right|_{s=0}\right)=2.12$$

其中，s 为 0 时刻的已有顾客数[参见文献 Kelton 和 law(1985)以及例 9.2]。表 9.2 给出了：500 个置信区间中估计覆盖到真实值 $d(25|_{s=0})$ 的比例，即估计覆盖度 $\hat{p}$，真覆盖度 p 值[即计算置信区间的数目非常大，它很可能覆盖 $d(25|_{s=0})$ 的比例]的 90%置信区间，以及置信区间半长的平均值[即 $t_{n-1,1-\alpha/2}\sqrt{S^2(n)/n}$]除以 500 次实验所得到的点估计值 $\overline{X}(n)$，该参数是置信区间精度的度量，进一步讨论见下面。真正覆盖度为 90%的置信区间由下式计算：

$$\hat{p}\pm z_{0.95}\sqrt{\frac{\hat{p}(1-\hat{p})}{500}}$$

它是基于这样一个事实，即$(\hat{p}-p)/\sqrt{\hat{p}(1-\hat{p})/500}$是近似服从标准正态分布的随机变量[参见，例如，文献 Hogg 和 Craig(1995，第 254-255 页)]。建议 p 的置信区间仅在 $n\hat{p}\geqslant10$ 和 $n(1-\hat{p})\geqslant10$ 时使用。如果在特殊情况下并非如此，那么该置信区间的值[Devore (2008，p266)]就很可能被使用了。

表 9.2　基于 500 次实验的 $d(25|s=0)=2.12$ 的固定样本长度结果，$\rho=0.9$ 的 $M/M/1$ 排队系统

n	估计的覆盖度 $\hat{p}$	(置信区间半长)的平均值/$\overline{X}(n)$
5	0.880±0.024	0.67
10	0.864±0.025	0.44
20	0.886±0.023	0.30
40	0.914±0.021	0.21

从表 9.2 可以看到，基于 $n=10$ 次重复运行的 500 个置信区间的 86.4% 覆盖 $d(25|s=0)$，并且我们有将近 90%的置信度相信 $n=10$ 时的真实覆盖度为 0.839 到 0.889 之间。考虑到仿真模型总是近似于其所对应的实际系统，所以我们认为表 9.2 给出的估计覆盖度已经足够接近之前希望的 0.9，从而是可用的。还要注意，从表 9.2 的最后一列可以看出，为了将置信区间的精度提高 1 倍，所需进行的重复运行次数大约为之前的 4 倍。这一点并不奇怪，因为在式(9.1)中，置信区间的半长表达式的分母上有一个$\sqrt{n}$。

为了说明由式(9.1)给出的置信区间并不总是得到接近 $1-\alpha$ 的覆盖度，我们考虑第二个例子。对于由三个部件组成的可靠性模型，系统能够正常工作的条件是，元件 1 能够正常工作，且元件 2 和元件 3 中的一个能够正常工作。如果 G 是整个系统的失效时间，G_i 表示部件 i 的失效的时间(其中，$i=1$，2，3)，则 $G=\min\{G_1，\max\{G_2，G_3\}\}$。我们进一步假设所有 G_i 为独立的随机变量，并且每个 G_i 都服从形状参数为 0.5、比例参数为 1 的韦布尔(Weibull)分布(参见第 6.2.2 小节)。这种特殊的韦布尔分布具有明显的不对称性和非正态性。我们再次运行 500 次独立仿真实验，对于每次试验，考虑 $n=5$、10、20、40，并且对于每个 n，利用式(9.1)构建 $E(G|\text{所有元件都为新的})=0.78$(通过解析推理计算得到)的 90%置信区间。表 9.3 给出了这些实验的结果。注意，当 n 值很小时，覆盖度

退化显著。而且，随着 n 不断增大，覆盖度逐渐接近 0.9，符合中心极限定理。

表 9.3 基于 500 次实验的 $E(G \mid$ 所有部件为新的$)=0.78$ 的固定样本长度法的结果，可靠性模型

n	估计的覆盖度 $\hat{p}$	置信区间半长的平均值/$\overline{X}(n)$
5	0.708±0.033	1.16
10	0.750±0.032	0.82
20	0.800±0.029	0.60
40	0.840±0.027	0.44

我们从表 9.2 和表 9.3 可以看出，由式(9.1)给出的置信区间实际所得到的覆盖度取决于所考虑的仿真模型，还取决于样本的数量 n。因此，人们自然会问，为什么 $M/M/1$ 排队模型中的置信区间好于可靠性模型。为了回答这个问题，我们首先对 $\rho=0.9$，$s=0(n=1)$的 $M/M/1$ 排队模型进行 500 次独立仿真实验，并在每次重复运行中观察 $\sum_{i=1}^{25} D_i/25$。500 个平均延误时间的直方图如图 9.5 所示，且样本偏度为 1.64(参见表 4.1 和表 6.5)。虽然该直方图表明平均延误时间并不是正态分布的，但是它的确说明平均延误时间的分布不是特别偏的(例如，指数分布的偏度为 2)。下面对可靠性模型进行 500 次独立实验，并观察在每次重复运行中的失效时间 G。G 的 500 个值的直方图在图 9.6 中给出，且估计的偏度为 3.64。所以，与平均延误时间的分布相比，失效时间的分布明显更具有非正态性。这些结果能够从一定程度上解释，与可靠性模型相比，为什么 $M/M/1$ 排队模型的覆盖度更接近 0.9。

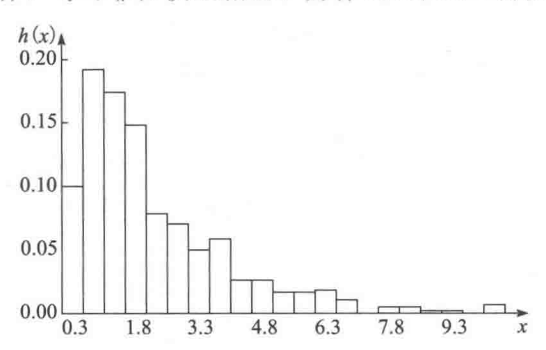


图 9.5 $\rho=0.9$ 的 $M/M/1$ 排队模型 500 个平均延误时间(每个基于 25 个单独延误)的直方图

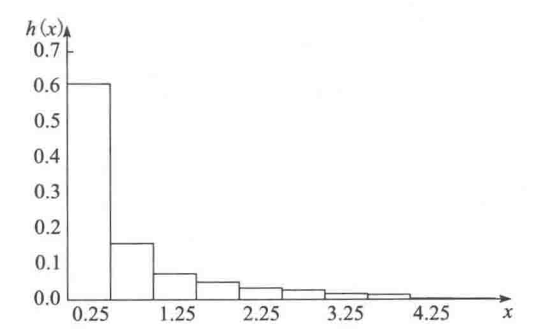


图 9.6 可靠性模型 500 个失效时间的直方图

读者可能奇怪为什么平均延误时间比失效时间更接近正态分布。注意，$M/M/1$ 排队模型中的 X_j 实际上是 25 个延误时间的平均值；而可靠性模型中的 X_j 是由三个失效时间按最小和最大公式计算得到的。对一类相关数据来说，中心极限定理指出，随着数据点个数平均增大，这些数据的均值逐渐逼近正态分布。为了对 $M/M/1$ 排队模型验证这一点，我们执行 500 次独立实验，并在每次重复运行中观察 $\sum_{i=1}^{6\,400} D_i/6\,400$。500 个平均延误时间(每个基于 6 400 个单独的延误时间)的直方图如图 9.7 所示，

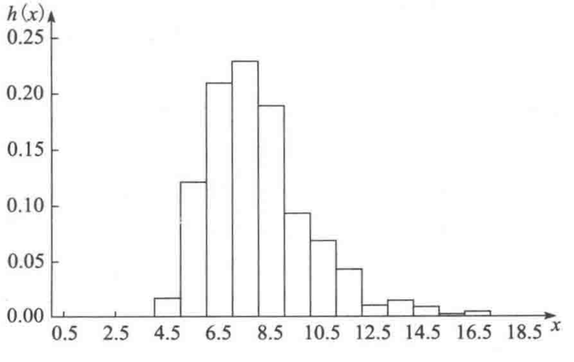


图 9.7 $\rho=0.9$ 的 $M/M/1$ 排队模型 500 个平均延误时间(每个基于 6 400 个单独的延误时间)的直方图

并且估计的偏度为 1.07（正态分布的偏度为 0）。显然，与图 9.5 所示的直方图相比，图 9.7所示的直方图更接近正态分布。

因此，我们可以期望，如果 X_j 为数量很大的单个观测值（即使这些观测值存在相关性）的均值，置信区间覆盖度的退化程度不会特别严重。经验表明，许多实际问题的仿真得到的 X_j 都属于这一类型。

获得指定的精度

基于 n 次重复运行的固定样本长度方法的一个缺点在于，分析者未控制置信区间的半长[或 $\overline{X}(n)$ 的精度]，对于给定的 n，置信区间的半长取决于 $\mathrm{var}(X)$，即 X_j 的总体方差。在接下来的内容中我们将讨论，为估计具有指定的误差或精度的均值 $\mu=E(X)$，如何确定所需重复运行次数的方法。

我们首先定义两种度量估计值 $\overline{X}$ 的误差的方法（不考虑对 n 的依赖关系，因为重复运行的次数是一个随机变量）。如果估计值 $\overline{X}$ 满足 $|\overline{X}-\mu|=\beta$，则我们称 $\overline{X}$ 的绝对误差为 β。如果我们不断重复运行仿真，直到根据式(9.1)得到的百分之 $100(1-\alpha)$ 置信区间的半长度小于或等于 β（其中 $\beta>0$），则有：

$$1-\alpha \approx P(\overline{X}-\text{半长} \leqslant \mu \leqslant \overline{X}+\text{半长}) = P(|\overline{X}-\mu| \leqslant \text{半长}) = P(|\overline{X}-\mu| \leqslant \beta)$$

如果 A 和 B 均为事件，并且 A 为 B 的子集，则有 $P(A)\leqslant P(B)$。因此，$\overline{X}$ 的绝对误差不超过 β 的概率近似等于 $1-\alpha$。换句话说，如果我们使用上面的停止规则构建 100 个独立的 90%置信区间，则我们可期望在这 100 个置信区间中大约有 90 个 $\overline{X}$ 的绝对误差不超过 β。

假定基于固定的重复运行次数 n 构建了 μ 的一个置信区间，如果设随着重复运行次数的增加，总体方差的估计 $S^2(n)$ 不变（稍微有一点），则为得到绝对误差 β 所需进行的重复运行次数 $n_a^*(\beta)$ 的近似表达式是：

$$n_a^*(\beta) = \min\left\{i \geqslant n : t_{i-1,1-\alpha/2}\sqrt{\frac{S^2(n)}{i}} \leqslant \beta\right\} \tag{9.2}$$

其中，冒号“:”读作“满足”。迭代地将 i 加 1，直到得到一个 i 值，满足 $t_{i-1,1-\alpha/2}\sqrt{S^2(n)/i}\leqslant\beta$，就能确定 $n_a^*(\beta)$ [另外一种办法，$n_a^*(\beta)$ 也可以近似为满足 $i\geqslant S^2(n)(z_{1-\alpha/2}/\beta)^2$ 的最小整数 i]。如果 $n_a^*(\beta)>n$ 且仿真的重复运行多进行 $n_a^*(\beta)-n$ 次，则基于所有 $n_a^*(\beta)$ 次重复运行得到的估计值 $\overline{X}$ 的绝对误差近似为 β。式(9.2)的精度取决于方差估计 $S^2(n)$ 与 $\mathrm{var}(X)$ 的接近程度。

例 9.17 对于例 9.14 的银行系统，假定我们希望估计期望的平均延误时间的绝对误差为 0.25 分钟且置信水平为 90%，由 10 次已有可用的重复运行可得到：

$$n_a^*(0.25) = \min\left\{i \geqslant 10 : t_{i-1,0.95}\sqrt{\frac{0.31}{i}} \leqslant 0.25\right\} = 16 \quad \blacktriangleleft$$

现在讨论另一种度量 $\overline{X}$ 的误差的方法。假设 $\mu\neq 0$，如果估计值 $\overline{X}$ 满足 $|\overline{X}-\mu|/|\mu|=\gamma$，则称 $\overline{X}$ 的相对误差为 γ，或者称 $\overline{X}$ 的百分比误差为百分之 100γ。假定不断进行仿真的重复运行，直到由式(9.1)给出的置信区间的半长与 $|\overline{X}|$ 之比小于或等于 $\gamma(0<\gamma<1)$，该比值为实际相对误差的估计，则：

$$\begin{aligned}
1-\alpha &\approx P(|\overline{X}-\mu|/|\overline{X}| \leqslant \text{半长}/|\overline{X}|) \\
&\leqslant P(|\overline{X}-\mu| \leqslant \gamma|\overline{X}|) && [(\text{半长}/|\overline{X}|)\leqslant\gamma] \\
&= P(|\overline{X}-\mu| \leqslant \gamma|\overline{X}-\mu+\mu|) && (\text{减 }\mu\text{ 之后再加上 }\mu) \\
&\leqslant P(|\overline{X}-\mu| \leqslant \gamma(|\overline{X}-\mu|+|\mu|)) && (\text{三角不等式}) \\
&= P((1-\gamma)|\overline{X}-\mu| \leqslant \gamma|\mu|) && (\text{代数变换}) \\
&= P(|\overline{X}-\mu|/|\mu| \leqslant \gamma/(1-\gamma)) && (\text{代数变换})
\end{aligned}$$

所以，$\overline{X}$ 的相对误差不超过 $\gamma/(1-\gamma)$ 的概率大约为 $1-\alpha$。换句话说，如果使用上面的停止规则构建 100 个独立的 90%置信区间，则我们可期望在这 100 个置信区间中大约有 90

个 $\overline{X}$ 的相对误差不超过 $\gamma/(1-\gamma)$，大约有 10 个的相对误差会大于 $\gamma/(1-\gamma)$。注意，我们得到的相对误差是 $\gamma/(1-\gamma)$，而不是所要求的 γ，因为是用 $|\overline{X}|$ 估计 μ。

再次假定基于固定的重复运行次数 n 构造了 μ 的置信区间，如果随着重复运行次数的增加，总体均值和总体方差不变(稍微有一点)，则相对误差达到 γ 所需进行的重复运行次数 $n_r^*(\gamma)$ 的近似表达式为：

$$n_r^*(\gamma) = \min\left\{ i \geqslant n : \frac{t_{i-1,1-\alpha/2}\sqrt{S^2(n)/i}}{|\overline{X}(n)|} \leqslant \gamma' \right\} \tag{9.3}$$

其中，$\gamma'=\gamma/(1+\gamma)$ 是为得到实际相对误差 γ 的“调整的”相对误差(同样地，$n_r^*(\gamma)$ 也可以近似为满足 $i \geqslant S^2(n)(z_{1-\alpha/2}/(\gamma'|\overline{X}(n)|))^2$ 的最小整数 i)。如果 $n_r^*(\gamma)>n$ 且我们多进行 $n_r^*(\gamma)-n$ 次仿真的重复运行，则基于所有 $n_r^*(\gamma)$ 次重复运行的估计值 $\overline{X}$ 的相对误差大约为 γ。

例 9.18 对于例 9.14 的银行系统，假定我们希望估计期望的平均延误时间具有相对误差 0.10 且置信水平为 90%，基于已有的 10 次重复仿真试验可以得到：

$$n_r^*(0.10) = \min\left\{ i \geqslant 10 : \frac{t_{i-1,0.95}\sqrt{0.31/i}}{2.03} \leqslant 0.09 \right\} = 27$$

其中，$\gamma'=0.1/(1+0.1)=0.9$ ◀

直接使用式(9.3)得到相对误差为 γ 的估计值 $\overline{X}$ 的困难在于 $|\overline{X}(n)|$ 和 $S^2(n)$ 也许不是它们所对应的总体参数的精确估计。如果 $n_r^*(\gamma)$ 大于实际需要的重复运行次数，则做重复运行的次数会很多，导致计算机资源的浪费。反之，如果 $n_r^*(\gamma)$ 太小，则基于 $n_r^*(\gamma)$ 次重复运行的估计 $\overline{X}$ 可能达不到我们想要的精度。为了得到具有指定的相对误差的 μ 的估计值且只做实际需要那么多次的重复运行，我们现在介绍一种序贯法(每次增加一次新的重复运行)。该方法假设 X_1，X_2，… 是一独立同分布的随机变量序列，但不必是正态分布的。

该方法的特定目标是得到的 μ 的估计具有相对误差为 $\gamma(0<\gamma<1)$，且置信水平为百分之 $100(1-\alpha)$。选择一个重复运行的初始次数 $n_0 \geqslant 10$，并令

$$\delta(n,\alpha) = t_{n-1,1-\alpha/2}\sqrt{\frac{S^2(n)}{n}}$$

为通常的置信区间的半长。则序贯法如下：

(1) 进行 n_0 次仿真的重复运行，并置 $n=n_0$。

(2) 由 X_1，X_2，…，X_n 计算 $\overline{X}(n)$ 和 $\delta(n, \alpha)$。

(3) 如果 $\delta(n, \alpha)/\overline{X}(n) \leqslant \gamma'$，则用 $\overline{X}(n)$ 作为 μ 的点估计并停止。等价地

$$I(\alpha,\gamma) = [\overline{X}(n) - \delta(n,\alpha), \overline{X}(n) + \delta(n,\alpha)] \tag{9.4}$$

为 μ 的具有所要求精度的近似百分之 $100(1-\alpha)$ 置信区间。否则，用 $n+1$ 替代 n，再做一次仿真的重复运行，并返回步骤(2)。

注意，每一次重复运行的结果获得后，该方法就计算 $\text{var}(X)$ 的一个新的估计，并且该方法中所需的重复运行的总次数为一个随机变量。

例 9.19 对于例 9.14 的银行，假定我们希望得到期望的平均延误时间的估计具有相对误差 $\gamma=0.1$ 且置信水平为 90%。利用以前的 $n_0=10$ 次重复运行作为起点，我们得到：

$$\text{终止时重复运行的次数} = 74$$
$$\overline{X}(74) = 1.76, \quad S^2(74) = 0.67$$
$$90\% \text{ 的置信区间}: [1.60, 1.92]$$

注意，实际所需的重复运行次数为 74，远远大于例 9.18 中所预测的次数 27，原因主要是基于 10 次重复运行的方差估计不精确。 ◀

虽然上述序贯法看起来很有吸引力，但问题自然产生了，即采用产生具有接近所要求的 $1-\alpha$ 的覆盖度的置信区间这种方法究竟有多好呢？Law，Kelton 和 Koeing(1981)指出，如果 $\mu\neq 0$[且 $0<\mathrm{var}(X)<+\infty$]，只要所要求的相对误差足够接近 0，则由式(9.4)得到的置信区间的覆盖度可以任意地接近 $1-\alpha$。通过对大量 μ 的真值已知的随机模型和概率分布进行采样(包括 $M/M/1$ 排队系统和上述的可靠性模型)，我们建议使用序贯法取 $n_0\geqslant 10$ 且 $\gamma\leqslant 0.15$。发现如果按照这些建议去做，对 90%置信区间来说，估计的覆盖度(基于每个模型进行 500 次独立实验)不小于 0.864。

与上面介绍的序贯法类似的是由 Chow 和 Robbins(1965)提出的序贯法，用于构建具有小绝对误差 β 的 μ 的百分之 $100(1-\alpha)$ 置信区间。而且，可以证明，倘若所要求的绝对误差足够接近 0，由该方法实际产生的覆盖度将任意接近 $1-\alpha$。然而，由于"绝对误差足够小"的含义与模型极其相关，而且由于 Law(1980)关于覆盖度的结果表明该方法对 β 的选择非常敏感，所以通常情况下我们不推荐使用 Chow 和 Robbins 的序贯法。

方法使用的建议

现在对终止型仿真的固定样本长度法和序贯法的使用给出一些建议。如果人们正在进行试探性的试验，那么置信区间的精度不会是特别重要，我们推荐使用固定样本长度法。然而，如果随机变量 X_j 是高度非正态的，且重复运行的次数 n 太小的话，则所构建的置信区间的实际覆盖度多少会小于所要求的。

从一个由 n 次重复运行组成的试探性试验中，人们可以估计每次重复运行的执行时间以及 X_j 的总体方差，然后由式(9.2)可得到为估计具有所要求的绝对误差 β 的 μ 所需要的重复运行次数的粗略估计 $n_a^*(\beta)$。另一种做法，人们由式(9.3)可得到为估计具有所要求的相对误差 γ 的 μ 所需要的重复运行次数的粗略估计 $n_r^*(\gamma)$。有的时候 β 和 γ 取值必须根据所需重复运行次数的执行时间来进行调整。如果最终决定构建小的相对误差 γ 的置信区间，则我们建议使用序贯法，取 $\gamma\leqslant 0.15$，$n_0\geqslant 10$。如果人们希望相对误差 γ 大于 0.15 的置信区间，则我们建议连续几次应用固定样本长度法。特别地，人们也许估计出 $n_r^*(\gamma)$，再搜集更多次重复运行的结果，比如说$[n_r^*(\gamma)-n]/2$ 次，然后基于现有的$[n_r^*(\gamma)+n]/2$ 次重复运行用式(9.1)构建一个置信区间。如果所得置信区间的估计相对误差依然大于 γ'，则可基于新的方差估计重新估计 $n_r^*(\gamma)$，并搜集必须增加的重复运行的某一部分的结果，以此类推。为构建具有小绝对误差 β 的置信区间，我们依然建议连续几次应用固定样本长度法。

不考虑每次重复运行的时间，我们建议至少要进行 3 到 5 次随机仿真的重复运行以估计随机变量 X_j 的变化程度。如果因为时间的原因不可能做到，则恐怕完全不应该进行仿真研究。

9.4.2　其他性能度量的估计

在本节中，我们讨论除均值之外的性能度量的估计。正如下面的例子表明的那样，用某一类平均系统响应来比较两个或多个系统有可能导致令人误解的结论。

例 9.20　考虑例 9.14 的银行，其中的利用率 $\rho=\lambda/(5\omega)=0.8$。我们将每个出纳员前面有一个队列并可换队的策略与所有出纳员面对一个队列的策略进行基于在队列中的期望平均延误时间(参见例 9.14)和基于在队列中的顾客的期望时间平均数进行比较，后者的定义如下：

$$E\left[\frac{\int_0^T Q(t)\,\mathrm{d}t}{T}\right]$$

其中，$Q(t)$表示 t 时刻队列中的顾客数；T 为银行的营业时间($T\geqslant 8$ 小时)。表 9.4 给出了对于每种策略运行一次仿真的结果[执行这些仿真运行以满足第 i 个顾客($i=1,2,\cdots,N$)的到达时间对两种策略来说是相同的，故开始服务的第 i 个顾客($i=1,2,\cdots,N$)的服务时

间对两种策略来说也是一样的]。因此，基于“平均系统响应”，就会出现两种策略是等效的。然而，显然不是这样。因为在多队列策略下，顾客不必按其到达的顺序接受服务，我们会期望这种策略导致顾客的延误时间有更大的可变性。由上面所用的相同的两次仿真运行结果进行计算，表 9.5 给出了两种策略下延误时间分别在区间[0, 5)(分钟)、[5, 10), …,[40, 45)的顾客数的期望比例(我们没有从这些运行估计方差，原因 4.4 节已经指出，即由相关的仿真输出数据计算出来的方差估计是有偏的)。从表 9.5 看出，采用多队列策略，顾客的延误时间恐怕会比采用单队列策略的长。特别地，如果一天内有 480 个顾客到达(期望值)，则在 5 个队列和单一队列两种策略下，延误时间大于或等于 20 分钟的期望顾客数分别等于 33 人和 6 人(对于更大的 ρ 值，两种策略的差异甚至会更大)。这个观察结果加上单队列策略具有更大的公平性也许已经使得很多机构，如银行、航空公司等，采用这种策略。

表 9.4 两种银行队列策略的仿真结果：均值

性能的度量	估计值	
	5 个队列	一个队列
期望的运行时间/小时	8.14	8.14
期望的平均延误时间/分钟	5.57	5.57
期望的队列中平均人数/人	5.52	5.52

表 9.5 两种队列策略的仿真结果：比例

区间/分钟	延误时间按区间的期望比例的估计	
	5 个队列	一个队列
[0, 5)	0.626	0.597
[5, 10)	0.182	0.188
[10, 15)	0.076	0.107
[15, 20)	0.047	0.095
[20, 25)	0.031	0.013
[25, 30)	0.020	0
[30, 35)	0.015	0
[35, 40)	0.003	0
[40, 45)	0	0

◀

从上面的例子可得出结论，在比较备选系统或者策略时，仅仅依据平均的系统行为特性有时可能导致令人误解的结论，进一步，而比例可能是一种有用的系统性能的度量。在例 9.16 中，我们说明了如何获得一个期望的比例的点估计和置信区间。在这一节中，我们说明如何对终止型仿真的相关概率和分位数进行类似的分析。

如 9.4.1 节中所述，令 X 为定义在某一次重复运行上的随机变量。假定我们希望估计概率 $p=P(X\in B)$，其中，B 为实数的集合。(例如，B 可能为例 9.20 中的区间$[20, +\infty)$)。进行 n 次独立重复运行，并令 X_1，X_2，…，X_n 为得到的独立同分布的随机变量。令 S 为 X_j 落到集合 B 的个数，则 S 服从参数为 n 和 p 的二项分布(参见 6.2.3 节)，且 p 的无偏点估计为：

$$\hat{p}=\frac{S}{n}$$

进一步地，如果 n“足够大”，则 p 的百分之 $100(1-\alpha)$ 置信区间为：

$$\hat{p}\pm z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

另一种方法参见文献 Welch(1983，第 285-287 页)以及习题 9.9。

例 9.21 对于例 9.14 的银行，假定希望得到如下概率的点估计值和近似 90%置信区间：

$$p = P(X \leqslant 15), X = \max_{0 \leqslant t \leqslant T} Q(t)$$

在这种情形下，$B=[0,\ 15]$。我们对银行仿真模型进行 100 次独立重复运行并得到 $\hat{p}=0.77$。这样，对于银行的每 100 天，有将近 77 天我们期望一天中最大队长不超过 15 个顾客，我们还可以得到 p 的近似 90%置信区间为：

$$0.77 \pm 0.07, \text{或者表示成} \quad [0.70, 0.84]$$ ◀

现在假定希望估计随机变量 X 分布的 q-分位数(第 $100q$ 个百分位数)x_q(定义参见第 6.4.3 小节)。例如 0.5-分位数就是中位数。如果 $X_{(1)}$，$X_{(2)}$，…，$X_{(n)}$ 为与 n 次独立重复运行所得 X_j 相应的序统计量，则 x_q 的点估计值就是样本的 q 分位数$\hat{x}_q$，它由下式得到：

$$\hat{x}_q = \begin{cases} X_{(nq)}, & nq \text{ 为整数} \\ X_{(\lfloor nq+1 \rfloor)}, & nq \text{ 不为整数} \end{cases}$$

令 r 和 s 为正整数，且满足 $1 \leqslant r < s \leqslant n$。如果 n “足够大”，则 x_q 的百分之 $100(1-\alpha)$置信区间可以由下式得到[参见文献 Conover(1999，第 143-148 页)]：

$$P(X_{(r)} \leqslant x_q \leqslant X_{(s)}) \geqslant 1-\alpha$$

其中，

$$r = \lceil nq + z_{\alpha/2}\sqrt{nq(1-q)}\, \rceil$$

且

$$s = \lceil nq + z_{1-\alpha/2}\sqrt{nq(1-q)}\, \rceil$$

如果 X 为连续随机变量，置信区间表达式中的大于或等于号变成等于号。关于分位数置信区间的进一步讨论可以在韦尔奇(1983，第 287-288 页)中找到。

例 9.22 对于例 9.14 的银行，假定我们希望确定等候大厅的面积以容纳排队等待的顾客。如果令 X 为例 9.21 中定义的最大队列长度，则我们可能希望建一个足够大的等候大厅，以容纳 $x_{0.95}$个顾客，即 X 的 0.95 分位数。根据前例的 100 次重复运行，我们得到 $\hat{x}_{0.95}=X_{(95)}=20$。因此，如果等候大厅能够容纳 20 名在队等待的顾客，则对于每 100 天，将近有 95 天大厅肯定够了。更进一步地，$x_{0.95}$ 的近似 90%置信区间为$[X_{(91)},\ X_{(99)}]=[19,\ 23]$(对本问题来说，$X$ 为离散随机变量，因此置信水平是近似的)。◀

对容差区间的讨论，有兴趣的读者还可以参考文献 Conover(1999，第 150-155 页)，容差区间是包含指定比例的随机变量 X 的值的区间(且具有某一指定的置信水平)。

9.4.3 初始条件选择

正如第 9.3 节所指出的，终止型仿真的性能的度量显然依赖于系统在 0 时刻的状态，因此，必须特别注意选取合适的初始条件。现举例来说明这个潜在的问题。假定我们希望估计中午 12 点至下午 1 点(最繁忙的时段)到达银行并完成其延误的所有顾客的期望平均延误时间。因为在中午银行恐怕是相当拥挤的，那么用顾客数为零来启动仿真(对排队系统进行仿真时通常采用的初始条件)，将引起期望平均延误时间的估计值偏低。现在我们讨论解决这个问题的两种启发式方法，其中第一种方法得到广泛应用(参见第 9.5.1小节)。

对于第一种方法，假设银行上午 9 点开始营业，这时没有顾客，那么我们可从上午 9 点没有顾客开始仿真，运行 4 个仿真小时。在估计所要求的期望平均延误时间时，我们只使用在中午 12 点至下午 1 点之间到达并完成其延误的那些顾客的延误时间。上午 9 点与中午 12 点之间(“预热期”)的过程决定了对中午仿真来说合适的条件。这种方法的一个缺点是，3 个小时的仿真时间并没有直接用于估计。结果，有人可能采取折中，从某个其他

时间，比如上午 11 点开始仿真，那时没有顾客。然而，无法保证在中午仿真的条件代表银行在中午的实际情形。

另一种方法是搜集若干中午银行中的顾客数的数据。令 $\hat{p}_i$ 为这些天中午顾客数等于 i 的($i=0$，1，…)天数所占的比例。然后，我们对银行进行从中午到下午 1 点的仿真，中午的顾客数从分布$\{\hat{p}_i\}$中随机选择(可假设在中午正在接受服务的所有顾客都是刚刚开始其服务。令所有服务在中午以新的服务开始是对银行中的实际情况的一种近似，因为中午正在接受服务的顾客可能已经部分完成了服务。然而，对于长度为 1 小时的仿真而言，这种近似的影响应该是可以忽略的)。

如果希望多次运行从中午到下午 1 点的仿真，则从$\{\hat{p}_i\}$中选取不同的样本进行每次运行。这种情况下，由这些运行得到的 X_j 仍然是独立同分布的，因为每次运行的初始条件都是从同一个分布中独立选取的。

9.5 稳态参数的统计分析

令 Y_1，Y_2，…为一个非终止型仿真一次运行的输出随机过程。假设当 $i\to+\infty$，有 $P(Y_i\leqslant y)=F_i(y)\to F(y)=P(Y\leqslant y)$，其中，$Y$ 为所关心的稳态随机变量，其分布函数为 F(这里在初始条件 I 上限制 F_i 的相关性)。如果 ϕ 为 Y 的一个特征值，例如 $E(Y)$，$P(Y\leqslant y)$，或者 Y 的分位数，则 ϕ 为稳态参数。估计 ϕ 的一个困难在于，$Y_i(i=1,2,\cdots)$ 的分布函数与 F 是不同的，因为，一般不能保证所选的 I 代表了“稳态行为特性”。这就导致基于观测值 Y_1，Y_2，…，Y_m 得到的 ϕ 的估计不具有“代表性”。例如，对于所有有限的 m，样本均值 $\overline{Y}(m)$为 $\nu=E(Y)$的有偏估计。我们刚才讨论的这个问题在仿真文献中称为初始瞬态问题(problem of the initial transient)或者启动问题(startup problem)。

例 9.23 为了更简洁地说明启动问题，考虑 $\rho<1$ 的 $M/M/1$ 排队系统中的延误时间 D_1，D_2，…的过程(见例 9.2)。根据排队理论，可以证明

$$P(D_i\leqslant y)\to P(D\leqslant y)=(1-\rho)+\rho[1-\mathrm{e}^{-(\omega-\lambda)y}],i\to+\infty$$

如果 0 时刻出现的顾客数 s 等于 0，则 $D_1=0$，且对于任意 i，有 $E(D_i)\neq E(D)=d$。另一方面，如果按照系统分布的稳态数选择 s[例子参见文献 Gross 和 Harris(1998，第 57 页)]，则对于所有 i，有 $P(D_i\leqslant y)=P(D\leqslant y)$且 $E(D_i)=d$(见习题 9.11)。因此，这种情况下不存在初始瞬态。 ◀

在实际中，稳态分布将是未知的，所以无法采用上面的初始化方法。下一节讨论处理实际中的启动问题的方法。

9.5.1 初始瞬态问题

假设我们希望估计稳态均值 $\nu=E(Y)$，通常情况下其定义为：

$$\nu=\lim_{i\to+\infty}E(Y_i)$$

也就是说，瞬态均值收敛到稳态均值。初始瞬态问题的最严重的结果恐怕就是对于任何 m，均有 $E[\overline{Y}(m)]\neq\nu$[进一步的讨论参见 Law(1983，第 1010-1012 页)]。处理这个问题最常建议的技术称为模型预热或者初始数据删除。其思想是删除运行初期的某些观测数据而只使用剩下的观测值来估计 ν。例如，已知观测值 Y_1，Y_2，…，Y_m，常常建议采用

$$\overline{Y}(m,l)=\frac{\sum_{i=l+1}^{m}Y_i}{m-l},1\leqslant l\leqslant m-1$$

而不是 $\overline{Y}(m)$作为 ν 的估计。通常情况下，人们期望 $\overline{Y}(m,l)$比 $\overline{Y}(m)$的偏差小，因为受初始条件选择的影响，靠近仿真的“开始”的观测值可能不能很好地具有稳态行为特性的代表性。例如，在 $s=0$ 的 $M/M/1$ 排队系统的情形，过程 D_1，D_2，…就确实如此，因为当

$i \to +\infty$时，$E(D_i)$单调递增到d(参见图9.2)。Fishman(1972)曾指出一阶回归[AR(1)]过程也是如此(参见第6.10.3小节)。

但是，有些作者曾质疑删除初始数据的有效性[例如，参见Grassmann(2011)]，所以我们首先更加细心地验证一下点估计$\overline{Y}(m, l)$。虽然，通常情况下相较于$\overline{Y}(m)$，$\overline{Y}(m, l)$的偏置较小，然而其方差较大，如Fishman(1972)曾指出一阶回归[AR(1)]过程。但是，点估计质量最常用的总度量(总体测度)可能是均方误差[参见Pasupathy和Schmeiser(2010)]。Blomqvist(1970)指出对于m足够大的$M/M/1$队列，l的值为0时可以最小化$\overline{D}(m, l)$的均方误差，定义为：

$$\text{MSE}[\overline{D}(m,l)] = E\{[\overline{D}(m,l) - d]^2\} = \{\text{Bias}[\overline{D}(m,l)]\}^2 + \text{var}[\overline{D}(m,l)]$$

其中，Bias表示有偏的。另一方面，Snell和Schruben(1979)和Kelton(1980)指出对于一阶回归AR(1)过程，基于m、l以及过程参数的值，删除既可以增加也可以减低均方误差。正如人们也许会怀疑的那样，当初始偏置很高并且自相关性很强时(参见章节5.6)，以至于偏置递减缓慢的情况下，删除最重要的作用是降低均方误差。在这些情况下，当m的值增大时，最小化均方误差$\text{MSE}[\overline{Y}(m, l)]$的$l$值则会下降。当$m$值非常大时，该观察值符合Blomqvist的结论，但并不建议将删除用于均方差性能度量。

另一种估计删除效能的标准，也是我们更为认可的一项标准，是置信区间质量[我们相信应该总是构建一个关于ν的置信区间；否则，我们没有一个确切的方法可以获知$\overline{Y}(m, l)$是如何趋近于ν的]。第9.5.2小节中曾讨论过，构建一个关于ν的置信区间的重复/删除方法，是基于令n独立“短”重复长度为m个观测值的Y_1，Y_2，…程序，并且每次重复运行时均删除前l个观测值的方法。令$\overline{Y}_j(m, l)$为第j次重复运行时删除前l个观测值后剩余的$m-l$个样本的均值，其中，$j=1, 2, \cdots, n$。然后，$\overline{Y}_j(m, l)$的功能等同于式(4.12)中所给出的置信区间中的X_j。为了在可接受的范围内产生置信区间的复制/删除方法，选择合适的m和l使得$E[\overline{Y}_j(m, l)] \approx \nu$是至关重要的，例如，$\overline{Y}_j(m, l)$是针对$\nu$的一个近似无偏估计值[参见Low(1977)]。

假设现在进行一次“长”运行得到m个观测值，分别为Y_1，Y_2，…，Y_m。基于这种重复一次的方案，构建一个关于ν的置信区间有许多种方法(参见9.5.3节)。例如，应用最广泛的批均值方法，将m个观察值分割成n批，每批的样本大小为$k(m=nk)$。令$\overline{Y}_j(k)$为第j批中k个观察值的样本(或批)均值，其中，$j=1, 2, \cdots, n$。然后，$\overline{Y}_j(k)$的功能等同于式(4.12)所给出的置信区间中的X_j。为了使批均值方法能够生成在可接受的覆盖范围内的置信区间，k值的选取必须足够大，使$\overline{Y}_j(k)$的值是近似不相关的。Low(1977)，Low和Carson(1979)以及Law和Kelton(1984)的结论显示，只要m的值“适中”，在无预热周期(例如，$l=0$)条件下，批均值(9.5.3节中的其他方法)能够生成在可接受的覆盖度内的置信区间。例如，对于$\rho=0.9$，$s=0$的$M/M/1$队列，基于$m=12\,800$和大小为2 560的$n=5$批样本下，d的批均值对于标称置信区间为90%的覆盖度达到了0.865[Low(1977)]。显然，如果m足够大，则保留“稳定状态”的观察值时，初始瞬值的观察值则会被“抹掉”。

因此，就置信区间覆盖度而言，当使用基于多个“短”重复的重复/删除方法时，一个有效的预热周期就显得非常重要。因此，我们将聚焦这种重复/删除方法，在接下来对初始瞬态问题的讨论中，需要对该方法进行无偏点估计。

问题自然就产生了，即如何选择预热周期(或删除个数)l。我们希望找到l(和m)使得$E[\overline{Y}(m, l)] \approx \nu$。如果$l$和$m$选得太小，则$E[\overline{Y}(m, l)]$有可能与$\nu$有很大的差异。另一方面，如果$l$比所需要的大，则$\overline{Y}(m, l)$可能有不必要的大方差。选择$l$的相关文献提供了很多方法。然而，Gafarian，Ancker和Morisaku(1978)发现当时可用的方法中没有一个在实际中表现得好。Kelton和Law(1983)开发了一种确定l(和m)的算法，该方法对范围广泛的随机模型工作得较好(也就是说，$E[\overline{Y}(m, l)] \approx \nu$)。然而，该方法存在一个理论

上的局限性，即它假设 $E(Y_i)$ 为 i 的单调函数。

确定 l 的最简单也最通用的一个方法是韦尔奇(1981，1983)提出的绘图方法。该方法的特定目标是确定一个时间索引点 l，使得 $i>l$ 时，有 $E(Y_i)=\nu$，这里的 l 就是预热周期[这等价于确定瞬态均值曲线 $E(Y_i)(i=1, 2, \cdots)$何时在水平 ν 变平，参见图 9.1]。一般而言，由于过程 Y_1，Y_2，…固有的可变性(参见图 9.8)，由一次重复运行来确定 l 是很困难的。因此，韦尔奇的方法是基于 n 次独立重复运行的仿真并采用以下 4 个步骤：

(1) 进行 n 次重复运行的仿真($n\geqslant 5$)，每次长度为 m(其中 m 很大)。令 Y_{ji} 为第 j 次重复运行的第 i 个观测值，如图 9.8 所示。

(2) 对于 $i=1, 2, \cdots, m$，令 $\overline{Y}_i=\sum_{j=1}^{n} Y_{ji}/n$(参见图 9.8)。平均过程 $\overline{Y}_1$，$\overline{Y}_2$，…的均值 $E(\overline{Y}_i)=E(Y_i)$，且方差 $\mathrm{var}(\overline{Y}_i)=\mathrm{var}(Y_i)/n$(参见习题 9.12)。所以，平均过程的瞬态均值曲线与原过程的相同，但它的图形只是方差的 $1/n$。

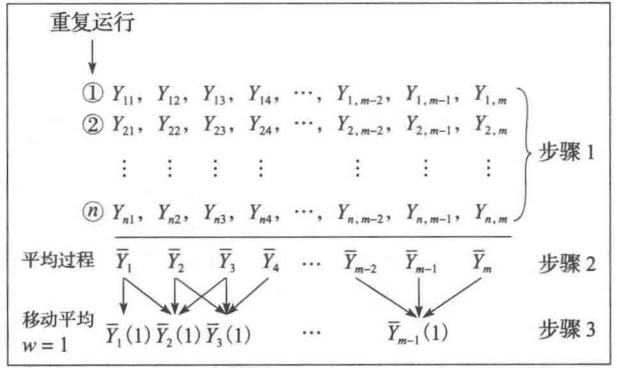


图 9.8 基于长度为 m 的 n 次重复运行的平均过程和 $w=1$ 的移动平均

(3) 为了过滤 Y_1，Y_2，…中的高频振荡(同时保留低频振荡或者所关注的长期趋势)，我们进一步定义移动平均 $\overline{Y}_i(w)$(其中，w 称为窗口，是一个正整数且满足 $w\leqslant\lfloor m/4\rfloor$)如下：

$$\overline{Y}_i(w)=\begin{cases}\dfrac{\sum\limits_{s=-w}^{w}\overline{Y}_{i+s}}{2w+1}, & i=w+1,\cdots,m-w\\[2ex] \dfrac{\sum\limits_{s=-(i-1)}^{i-1}\overline{Y}_{i+s}}{2i-1}, & i=1,\cdots,w\end{cases}$$

因此，如果 i 不是太靠近重复运行的开始，则 $\overline{Y}_i(w)$就是以第 i 个观测值为中心的平均过程的 $2w+1$ 个观测值的简单平均(参见图 9.8)。这称为移动平均，因为 i 随时间移动。

(4) 对于 $i=1, 2, \cdots, m-w$ 绘制 $\overline{Y}_i(w)$的图形，取 l 等于如下的 i 值，即 i 以后 $\overline{Y}_1(w)$，$\overline{Y}_2(w)$，…出现收敛趋势。关于帮助确定收敛的方法，参见韦尔奇(1983，第 292 页)。

下面的例子对移动平均的计算加以说明。

例 9.24 为了简单起见，假设 $m=10$，$w=2$，当 $i=1, 2, \cdots, 5$ 时，$\overline{Y}_i=i$，当 $i=6, 7, \cdots, 10$ 时，$\overline{Y}_i=6$，则

$$\overline{Y}_1(2)=1,\quad \overline{Y}_2(2)=2,\quad \overline{Y}_3(2)=3$$
$$\overline{Y}_4(2)=4,\quad \overline{Y}_5(2)=4.8,\quad \overline{Y}_6(2)=5.4$$
$$\overline{Y}_7(2)=5.8,\quad \overline{Y}_8(2)=6$$ ◀

在给出对实际的随机模型应用韦尔奇方法的例子之前，我们先给出一些选择参数 n、m 和 w 的建议。

- 进行 $n=5$ 或 10 次重复运行(取决于模型执行时间)，m 的大小要实际。特别地，m 应当远远大于 l 的预期值(参见第 9.5.2 小节)，并且大到足以允许不常见事件(例如机器故障停机)出现合理的次数。
- 绘制若干个窗口 w 的 $\overline{Y}_i(w)$，并选择使得相应的图形“较为平滑”的最小的 w 值(如果存在)。使用该图确定预热周期的长度 l[选择 w 的方法类似于直方图选择区间

Δb 长度的方法(参见第 6.4.2 小节)。如果 w 值太小，$\overline{Y}_i(w)$ 的图形会比较粗糙。如果 w 值太大，$\overline{Y}_i$ 的观测值就会过于聚集而使得我们不好考虑瞬态均值曲线 $E(Y_i)$ $(i=1, 2, \cdots)$ 的形状]。

- 如果第(3)步中不存在满足条件的 w 值，则再进行 5 次或 10 次长度为 m 的重复运行。使用所有可用的重复运行重复步骤(2)[对于一个固定的 w，随着重复运行次数的增多，$\overline{Y}_i(w)$ 的图形将会变得更平滑。请读者考虑其原因]。

在实际中应用韦尔奇方法的一个主要困难是，如果过程 Y_1，Y_2，…的波动大，所需的重复运行次数 n 相对也会大。另外，l 的选择有某种程度的主观性。

例 9.25 一个小型工厂由串行的一个加工中心和一个检测站组成，如图 9.9 所示。未完成的工件到达工厂的间隔时间服从均值为 1 分钟的指数分布。在机床上的加工时间服从[0.65，0.70]分钟的均匀分布，后续的在检测站的检测时间服从[0.75，0.80]分钟的均匀分布。所有被检测的工件中，有 90%是“合格的”并送去包装，10%的工件“不合格”并送回到机床重新加工(假设两个队列都有无限容量)。加工中心随机发生故障停机。特别地，一个新的(或者刚维修好的)机床在日历时间平均 6 小时的指数分布数量的时间后将会停机(参见第 14.4.2 小节)，维修时间是[8，12]分钟区间上的均匀分布。如果机器停机时有工件正在加工，则维修完成后，继续完成剩余的加工任务。假设工厂最初空且闲，并且每天工作 8 小时。

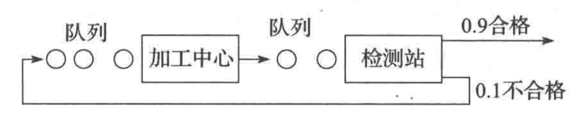


图 9.9 由一个加工中心和一个检测站组成的小型工厂

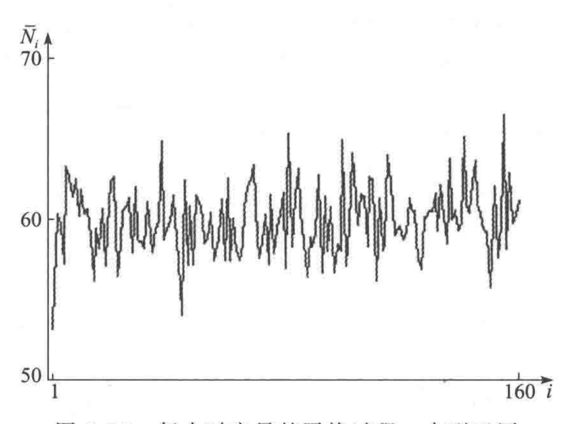


图 9.10 每小时产量的平均过程，小型工厂

考虑随机过程 N_1，N_2，…，其中 N_i 表示第 i 个小时生产的工件数。假设我们需要确定预热周期 l 使得我们最终能估计稳态平均每小时的产量 $\nu=E(N)$(参见例 9.28)。我们进行 $n=10$ 次独立重复运行仿真，每次长度 $m=160$ 小时(或 20 天)。在图 9.10 中画出了该平均过程 $\overline{N}_i(i=1, 2, \cdots, 160)$ 的图形。显然，该图形需要进一步平滑，而且，通常情况下，一次重复运行不足以估计 l。在图 9.11a 和 9.11b 中，我们画出了 $w=20$ 和 $w=30$ 两者的移动平均 $\overline{N}_i(w)$ 的图形。根据 $w=30$ 时的图形(它更为平滑)，我们选择预热周期 $l=24$ 小时。注意，l 太大优于太小，因为我们的目标是使得 $i>l$ 时，$E(Y_i)$ 接近 ν(我们选择允许稍高一些的方差是为了更确定地使 ν 的点估计会有小的偏差)。

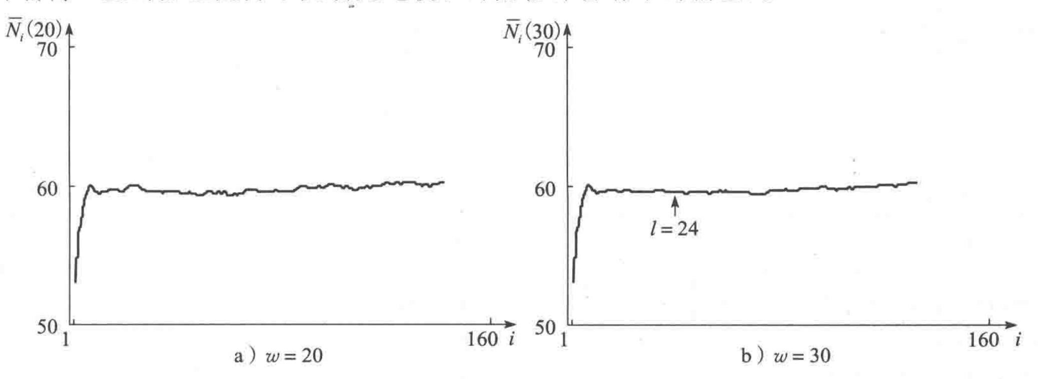


图 9.11 每小时产量的移动平均，小型工厂 ◀

例 9.26 考虑7号信号系统(signaling system number 7，SS7)网络的简单模型，用于连接和切断电话呼叫，以及处理“800”呼叫(实际的呼叫是通过一个相关的开关电路网传输的)。该网络包括4个信号节点(依次记作SP-1，…，SP-4)、两对信号传输节点(STP-A/STP-B和STP-C/STP-D)、传输速率为56kb/s、双工(双向)链路对(参见习题9.33)，如图9.12所示(图9.12中的一个线段对应两条链路)。从节点SP-1到节点STP-A的链路记作1-A，从STP-A到STP-C的链路记作A-C，等等。系统要求每个SP和链路的利用率不得超过0.4(在实际的网络中，这个要求是必需的，是为了允许额外的能力来防止某个资源故障中断；不过，我们这里并不对故障中断建模)。

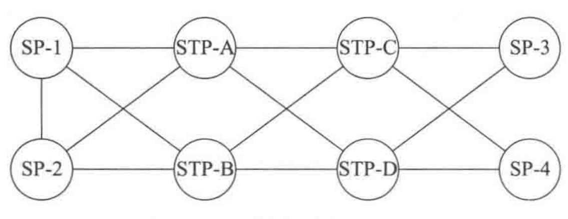


图 9.12 SS7 网络的拓扑

注：每个线段表示两条链路

每个SP按泊松过程(即指数到达间隔时间)给其他每个SP发送报文(信号)，速率在表9.6中给出。每个报文的长度是离散均匀随机变量，范围为23字节到29字节。当报文在链路上发送时，每个报文还包含一个7字节的路由标签(包含源节点和目标节点)

表 9.6 一个SP到另一个SP的传输速率(每分钟的报文数)

节点	SP-1	SP-2	SP-3	SP-4
SP-1		9 600	7 200	4 800
SP-2	8 000		4 800	7 200
SP-3	6 400	4 800		6 400
SP-4	4 800	5 600	4 800	

◀

每个STP(SP)有3个(2个)并行处理器(参见习题9.34)，由单一输入队列馈送，且源节点的每个链路都有一个输出队列。报文必须由节点中的一个处理器处理，处理时间为常数2.5毫秒。

表9.7给出了用于由一个节点向另一节点发送报文的初始链路，当两个链路均可用时，则每个选中的概率为0.5。

表 9.7 由一个节点(行)到另一个节点(列)的(见图9.12)初始链路

节点	SP-1	SP-2	SP-3	SP-4
SP-1		1-2	1-A，1-B	1-A，1-B
SP-2	1-2		2-A，2-B	2-A，2-B
SP-3	3-C，3-D	3-C，3-D		3-C，3-D
SP-4	4-C，4-D	4-C，4-D	4-C，4-D	
节点	**SP-1**	**SP-2**	**SP-3**	**SP-4**
STP-A	1-A	2-A	A-C，A-D	A-C，A-D
STP-B	1-B	2-B	B-C，B-D	B-C，B-D
STP-C	A-C，B-C	A-C，B-C	3-C	4-C
STP-D	A-D，B-D	A-D，B-D	3-D	4-D

考虑随机过程 E_1，E_2，…，其中，E_i 表示第 i 个完成的报文端到端的延迟时间(即从源SP到目的SP的时间)。假定我们希望确定预热周期使得我们最终能估计稳态均值 $\nu=E(E)$(参见例9.29)[符号 $E(E)$是稳态随机变量 E 的期望值]。我们进行 $n=5$ 次独立重复运行仿真，每次长度 $m=10$ 秒。在图9.13中我们绘制了 $w=600$ 的端到端延迟时间的移

动平均 $\overline{E}_i(w)$的图形[注意，在 10 秒的仿真运行中，E_i 观测值的个数是一个均值近似为 12 400 的随机变量，因为总的到达速率是 1 240 个报文每秒，并且系统是稳定的(参见习题 9.35)。因此，对我们的分析来说，我们使用了 5 次运行的任何一次的最小个数的观测值，最少个数为 12 306。但是图 9.13 中仅绘制了 $i=1$, 2, …, 9 920(1 240 的倍数)的$\overline{E}_i(w)$]。由该图，我们保守地选取预热周期 $l=$ 6 200×(5×1 240)端到端延迟时间。然而，为构建例 9.29 中的置信区间，我们实际将使用 5 秒的预热周期，因为运行长度 m 是以秒为单位的。

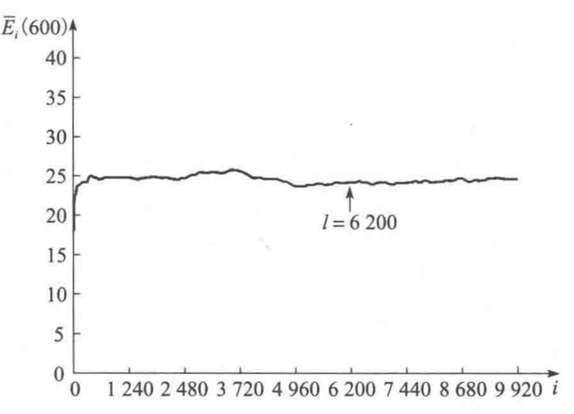


图 9.13　端到端延迟时间 $w=600$ 的移动平均 $\overline{E}_i(w)$，SS7 网络

例 9.27 考虑例 9.3 库存系统的过程 C_1, C_2, …。假定我们希望确定预热周期 l 以估计稳态每月平均成本 $c=E(C)=112.11$。我们进行 $n=10$ 次独立重复运行仿真，长度 $m=100$ 个月。在图 9.14 中，我们绘制了 $w=20$ 的移动平均 $\overline{C}_i(w)$的图形，由此我们选择预热周期 $l=30$ 个月。◀

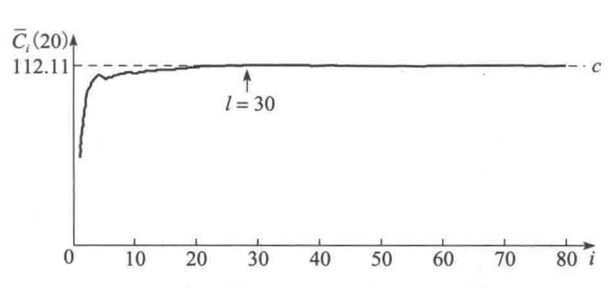


图 9.14　每月成本的 $w=20$ 的移动平均 $\overline{E}_i(w)$，库存系统

第 10～13 章给出了韦尔奇法的更多应用。还要注意，在制造系统仿真软件包 AutoMod 中也使用了韦尔奇法[见 Banks(2004)]。

White(1997)介绍了一种基于最小化均方误差确定预热周期或删除数量的过程，称为 MESR(边缘标准误差规则)。令 $\overline{Y}(m, l)$如以上定义，在不选择 $m-l-1$ 而选择$m-l$作为分母的情况下，令

$$S^2(m,l)=\frac{\sum_{i=l+1}^{m}[Y_i-\overline{Y}(m,l)]^2}{m-l}$$

为 Y_{l+1}, Y_{l+2}, …, Y_m 的样本方差。定义 MESR(m, l)统计量为

$$\mathrm{MESR}(m,l)=\frac{S^2(m,l)}{m-l}$$

如果 Y_i 是独立同分布的，$S^2(m, l)$中的分母为 $m-l-1$，则 MESR(m, l)统计量为 $\overline{Y}(m, l)$方差的无偏估计[$\overline{Y}(m, l)$标准误差的平方]，则可选的删除数量 l^* 为使得 MESR(m, l)取最小值时所对应的 l 值，其中，$l=0$, 1, …, $m-1$，通常记为：

$$l^*=\underset{l=0,1,\cdots,m-1}{\arg\min}\ \mathrm{MSER}(m,l) \tag{9.5}$$

其中，“arg”为参数的缩写。

Pasupathy 和 Schmeiser(2010)指出对于所有的 l，MESR(m, l)渐进地(当 m 趋近于无穷时)与均方误差成正比。因此，对于较大的数 m，使得 MESR(m, l)取最小值的 l 值将趋向于接近使得 MESR[$\overline{Y}(m, l)$]取最小值的 l 值。虽然 MSER 明显可以最小化均方误差，但是它的分量也说明它是产生 $\overline{Y}(m, l)$中偏差的一个步骤[参见 Hoad 等(2009，第 9 页)和 Franklin 和 White(2008，第 545 页)]。

White 等(2000)讨论了 MSER 的一种变体，称为 MSER-k。令

$$Z_j = \frac{\sum_{i=1}^{k} Y_{k(j-1)+i}}{k}, j = 1, 2, \cdots, \lfloor m/k \rfloor$$

则 MSER-k 是式(9.5)中给出的规则应用于 Z_j 的批平均值，而不是 Y_i 的批平均值，当然，其中，MSER-1 等同于 MESR。实际上，较为常用的是 MESR-5 而不是 MESR，因为前者是在“平滑”的数据上进行操作的。此外，如果 $l^* > \lfloor m/k \rfloor/2$(批数量的一半)，则 l^* 不能作为有效的预热周期。在这种情况下，应该在增大 m 的值同时将 MESR-5 用于新的批平均值集合中，等等。注意，选取 $k=5$ 以及$\lfloor m/k \rfloor/2$ 或许有些武断，但是在实际应用中的效果却非常好。最后，Hoad 等(2009)曾评论称，MESR-5 适用于平均重复运行次数超过 5 次以上的数据而不适用于通过一次试验所获得的数据(参见例 9.28)。此外，Mokashi 等(2010)以及 Sanchez 和 White(2011)也讨论过有关 MESR-5 的问题。

例 9.28 考虑 $\rho=0.9(\lambda=1, \omega=10/9)$，初始条件为 $s=0$ 的 $M/M/1$ 排队模型中的在队列中的延误时间过程 D_1，D_2，…。我们对长度为 $m=65\,000$ 的观察值进行 $n=5$ 次独立重复运行，并在平均过程 $\overline{D}_1$，$\overline{D}_2$，…，$\overline{D}_{65\,000}$ 中应用 MESR-5 导致了总删除数量达到 15 个观察值(所有的计算都是使用华威大学的 Katy Hoad 教授提出的 Excel 的宏指令进行的；允许 m 可达到的最大值为 65 536)。依据在本节前面所讨论的布洛姆奎斯(Blomqvist)给出的结果，l^* 接近于 0 的事实并不奇怪。图 9.15 给出了批数函数 MESR-5 统计量与 l 的关系，其中可以得出，当 $l=3$ 时，MESR-5 统计量达到最小值。同时从图 9.2 中还可以得出 MESR-5 不能够删除大量的偏置数据。

为了观察最佳删除数量 l^* 可能对 m 的值有多敏感，我们使用已存在的 65 000 个观察值的子集重复上述分析，子集的样本大小分别为 1 000、5 000、10 000、20 000 和40 000。对于这 5 个子集的样本大小，l^* 的值均为 3 或 4。

为了观察批样本大小 k 对于选择预热周期的影响，我们应用 MESR-1(例如，$k=1$)对长度为 $m=65\,000$ 的样本进行 $n=5$ 次重复运行，并且满足 $l^*=16$，导致了总删除数量达到 16 个观察值(同上述删除总数量为 15 的情况进行比较)。 ◀

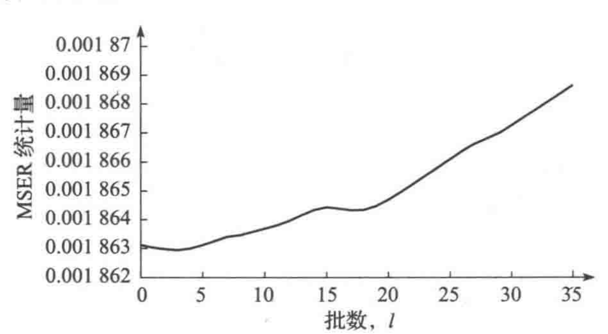


图 9.15 $\rho=0.9$ 的 $M/M/1$ 队列，批数函数 MESR-5 统计量与 l 的关系

例 9.29 再次考虑例 9.26 中的 SS7 网络，其中我们对长度 $m=12\,306$的观察值进行 $n=5$ 次独立试验。我们将 MESR-5 应用于平均过程 $\overline{E}_1$，$\overline{E}_2$，…，$\overline{E}_{12\,306}$，满足 $l^*=2$ 批的预热周期，导致了总删除数量为 10 个观察值(回顾例 9.26 中韦尔奇的程序要求的 6 200 个观察值的预热周期)。图 9.16 给出了批数函数 MESR-5 统计量与 l 的关系，从中可以得出：当 $l=2$ 时可以最小化 MESR-5 统计量。图 9.13 显示 MESR-5 不能删除大量偏置数据。

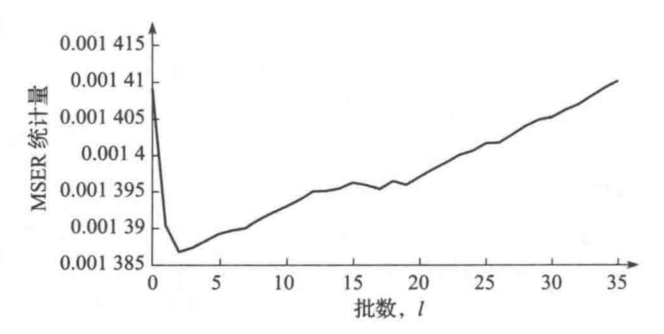


图 9.16 SS7 网络，批数函数 MESR-5 统计量与 l 的关系

为了观察最佳删除数量 l^* 可能对 m 值的敏感度，我们使用已存在的

12 306个观察值的子集重复上述分析，子集的样本大小分别为 1 000、2 000、4 000 和 8 000。对于这 4 个子集样本，l^* 值均为 2。◀

基于上述给出的两个例子(以及其他未给出的示例)，很显然对于某些仿真模型，MESR 也许不能删除大量的高偏置数据。

Schruben(1982)研究出了一种基于标准化时间序列的非常通用的方法(见第 9.5.3 小节)，以确定观测值 Y_{s+1}，Y_{s+2}，…，Y_{s+t}(其中 s 不必是 0)是否对稳态均值 $\nu=E(Y)$ 含有初始偏差，即是否至少存在一个 i，使得 $E(Y_i)\neq\nu$ 成立(其中 $s+1\leqslant i\leqslant s+t$)。当建立这个方法时，它并不是为确定删除数量 l 的算法，而是用来检验一组观测值是否含有初始偏差。例如，可以使用该方法将来自应用韦尔奇法的平均过程 $\overline{Y}_{l+1}$，$\overline{Y}_{l+2}$，…，$\overline{Y}_m$ 进行截断，以便确定是否还存在明显的剩余偏差。Schruben 对多个已知 ν 值的随机模型进行过该方法的检验，并发现该方法具有很强的探测初始化偏差的能力[也可参见 Glynn(1995)]。Schruben，Singh 和 Tierney(1983)以及 Goldsman，Schruben 和 Swain(1994)给出了一些这种初始化偏差检验方法的变化形式。最后，Vassilacopoulos(1989)提出了一种秩检验方法用于评估是否存在初始化偏差，对 $M/M/s$ 队列的有限检验得到了很理想的结果。

在例 9.23 中，我们看到，使用系统分布中的稳态顾客数来对 $M/M/1$ 队列初始化，得到的输出过程 D_1，D_2，…就没有初始瞬态。这就提示我们试图用一次“控制”性运行来估计稳态分布，然后对这个估计分布独立采样，以便为每次输出运行确定初始条件。Kelton(1989)将这个思想应用到了一些排队系统和一些计算机模型中，其中每一种情形，系统的状态是整数值随机变量。他发现，与用固定状态(如，排队系统中没有任何顾客)启动仿真相比，随机初始化减小了初始瞬态周期的敏感度和持续时间。但是，这种方法可能难以应用到很多实际仿真的情况中，因为实际系统的状态是多变量分布[更进一步讨论参见 Murray(1988)和 Law(1983，第 1016 页)]。Glynn(1988)讨论了一种相关的方法，该方法使用一次通过“瞬态周期”的做法来指定后续重复运行的启动条件。最终，有关初始瞬时问题的全部参考文献可参见 Hoad 等(2009)以及 Pasupathy 和 Schmeiser(2010)。

9.5.2 均值的重复运行/删除法

假定希望估计过程 Y_1，Y_2，…的稳态均值 $\nu=E(Y)$。有 6 种基本的方法用于解决这个问题，这些方法将在本节和下一节中讨论。但是，我们大部分将着重其中的一种——重复运行/删除法，原因如下：

(1) 如果应用合理，该方法应合理地给出良好的统计性能。

(2) 它是最容易理解和实现的方法(这一点在实际中是非常重要的，因为很多仿真项目有时间约束，而且很多研究人员不具备使用某些更复杂的分析方法所必需的统计学知识)。

(3) 该方法适用于所有类型的输出参数(即第 9.4 节～9.6 节)。

(4) 它可以容易地用于估计同一仿真模型的多个不同参数。

(5) 该方法可用于比较不同的系统配置，正如第 10 章中讨论的。

(6) 多个重复运行可以在通过局域网连接的多台计算机上同时做(仿真软件包 AutoMod 有这个能力)。

现在我们给出重复运行/删除法以获得 ν 的点估计和置信区间。分析方法类似于终止型仿真，这里在每次重复运行中，只使用那些过了预热周期 l 的观测值来形成估计值。特别地，假定我们进行了 n'次重复仿真，每次长度为 m'个观测值，这里 m'远远大于由韦尔奇的图形方法确定的预热周期 l(参见第 9.5.1 小节)。令 Y_{ji} 的定义如前，并令 X_j 由下式给出：

$$X_j=\frac{\sum_{i=l+1}^{m'}Y_{ji}}{m'-1},j=1,2,\cdots,n'$$

注意，X_j 只用到那些与“稳态”相对应的第 j 次重复运行的观测值，即 $Y_{j,l+1}$，$Y_{j,l+2}$，…，

$Y_{j,m'}$。因此所有 X_j 为独立同分布的随机变量，$E(X_j)=\nu$(参见习题 9.15)，$\overline{X}(n')$ 为 ν 的近似无偏估计，且 ν 的近似百分之 $100(1-\alpha)$ 置信区间为：

$$\overline{X}(n') \pm t_{n'-1,1-\alpha/2}\sqrt{\frac{S^2(n')}{n'}} \tag{9.6}$$

其中，$\overline{X}(n')$ 和 $S^2(n')$ 是分别根据式(4.3)和式(4.4)计算得到的。

反对重复运行/删除法所提出的一个合理理由是该方法使用了一组 n 次重复运行(控制运行)来确定预热周期 l，而然后只利用不同组的 n' 次重复运行(输出运行)的后 $m'-l$ 个观测值来进行实际的分析。然而，这通常不是问题，因为计算时间的费用相对较低。

在某些情况下，应该有可能使用长度为 m 个观测值的初始 n 个控制运行既确定 l，又构建置信区间。特别地，如果 m 远远大于所选预热周期 l 的值，则将“初始”运行用于两个目标大概也是安全的。因为韦尔奇图形法只是一种近似方法，预热期 l 之后的观测值个数少相对于 ν 会有明显的偏差。然而，如果 m 远远大于 l，则这些带有偏差的观测值对 X_j(基于 $m-l$ 个观测值)的总体质量(即去偏)或 $\overline{X}(n)$ 的影响就很小。但是，严格地说，基于两组独立重复运行的重复运行/删除法在统计上更正确一些(参见习题 9.1 和例 9.29)。

例 9.30 对于例 9.25 中的制造系统，假定我们希望得到稳态平均每小时产量 $\nu=E(N)$ 的点估计及 90%置信区间。根据例 9.25 中所使用的 $n=10$ 次长度 $m=160$ 小时的重复运行，我们指定预热周期 $l=24$ 小时。因为 $m=160$ 远大于 $l=24$，所以将使用这些同样的重复运行来构建置信区间。令

$$X_j = \frac{\sum_{i=25}^{160} N_{ji}}{136}, j = 1,2,\cdots,10$$

则 ν 的点估计和 90%置信区间分别为：

$$\hat{\nu} = \overline{X}(10) = 59.97$$

$$\overline{X}(10) \pm t_{9,0.95}\sqrt{\frac{0.62}{10}} = 59.97 \pm 0.46$$

因此，在长期运行中，我们可期望小型工厂的产量大约为每小时 60 个工件，这个产量合理吗(参见习题 9.17)? ◀

例 9.31 对于例 9.26 中的 SS7 网络，假定希望得到稳态平均端到端的延迟时间 $\nu=E(E)$ 的点估计和 95%置信区间。对于这个例子，我们进行 $n'=5$ 次长度为 $m'=65$ 秒的新的独立重复仿真，并使用之前确定的预热周期 $l=5$ 秒。令 X_j 表示第 j 次重复运行的完成时间在区间[5, 65]上的所有报文的平均端到端的延迟时间，则 ν(单位为毫秒)的点估计和 90%置信区间分别为：

$$\hat{\nu} = \overline{X}(5) = 24.11$$

$$\overline{X}(5) \pm t_{4,0.975}\sqrt{\frac{0.0114}{5}} = 24.11 \pm 0.13$$

由这 5 次重复运行，我们还发现每个 STP 和链路的利用率小于 0.4，正如所期望的那样。特别地，STP-A 的利用率是 0.316，链路 1～链路 2 的利用率为 0.377，这些值合理吗(参见习题 9.36)? ◀

式(9.5)给出的重复运行/删除法置信区间的半长取决于 X_j 的方差 $\mathrm{var}(X_j)$，而在进行最初的 n 次仿真试验时这个数值是未知的。因此，如果我们将重复仿真的次数进行固定，得到的置信区间的半长的大小对特定的目标来说也许够也许不够。然而，我们知道，通过把重复运行次数变成 4 倍，置信区间的半长可缩短到原来的一半。也可参见第 9.4.1 小节中“获得指定精度”的讨论。

关于重复运行/删除法有时有这样的批评，即实际构建的是 $E(X_j)$ 百分之 $100(1-\alpha)$ 置信区间而不是 ν 的[即 $\overline{X}(n')$ 为 ν 的一个有偏估计]。因此，如果为了缩小置信区间的半长而进行了大量的重复运行，则置信区间的覆盖度将会远小于所希望的 $1-\alpha$。然而，因为仿真模型往往只是相应真实系统的一个近似，所以，我们认为，对于很多(如果不是大多数的话)模型而言，倘若 $E(X_j)$ “接近” ν，估计 $E(X_j)$ 就已经足够了。要想满足这个条件，如果我们选择的运行长度 m' 足够大，并使用韦尔奇法来确定一个保守的预热周期 l 的话，应该就是这种情况。

9.5.3　均值的其他方法

在本节中我们对构建仿真输出过程 Y_1，Y_2，… 的稳态均值 $\nu=E(Y)$ 的点估计和置信区间的方法进行更全面的讨论。通常情况下，下面两个关于 ν 的定义是等价的：

$$\nu=\lim_{i\to+\infty}E(Y_i)$$

$$\nu=\lim_{m\to+\infty}\frac{\sum_{i=1}^{m}Y_i}{m}\quad(\text{w. p. }1)$$

关于这个问题的参考文献有：Alexopoulos，Goldsman 和 Serfozo(2006)、Banks 等(2010)、Bratley，Fox 和 Schrage(1987)、Fishman(1978，2001)、Law(1983)和 Welch(1983)。

为了构建 ν 的点估计和置信区间，仿真文献中提到了两种通用的方法：

(1) 固定样本长度法。进行单一的任意的固定长度的仿真，然后根据可用数据，采用许多可用的方法之一来构建置信区间。

(2) 序贯法。单一的仿真运行的长度序贯地增加，直到能构建出“可接受的”置信区间为止。能用若干种方法确定何时停止仿真运行。

固定样本长度法

文献中一共提出了 6 种固定样本长度法[相关的综述参见 Law(1983)以及 Law 和 Kelton(1983)]。第 9.5.2 小节讨论了重复运行/删除法，它是基于长度为 m 个观测值的 n 次独立“短”重复运行。它有点估计 $\hat{\nu}$ 存在偏的趋势(参见第 9.1 节)。另外 5 种方法都是基于一个“长”重复运行的，它们有点估计 ν 的方差的估计值 $\widehat{\text{var}}(\hat{\nu})$ 存在偏问题的趋势。表 9.8 给出了 6 种方法的特性，而 5 种新方法的细节下文也会给出。

表 9.8　稳态估计方法的特性

方法	重复运行次数	最严重的偏问题	潜在困难
重复运行/删除法	$n(n\geqslant 2)$	$\hat{\nu}$	预热周期 l 的选择
批均值法	1	$\widehat{\text{var}}(\hat{\nu})$	为获得不相关批均值的批量大小 k 的选择
自回归法	1	$\widehat{\text{var}}(\hat{\nu})$	自回归模型的质量
谱分析法	1	$\widehat{\text{var}}(\hat{\nu})$	协方差滞后数 q 的选择
再生法	1	$\widehat{\text{var}}(\hat{\nu})$	存在“小”平均长度的周期
标准时间序列法	1	$\widehat{\text{var}}(\hat{\nu})$	批量大小 k 的选择

批均值法与重复运行/删除法相似，设法得到独立的观测值，从而利用第 4 章中的公式来得到置信区间。然而，因为批均值法是基于单一的长运行的，所以它必须经历一次“瞬态周期”。假设 Y_1，Y_2，… 为协方差平稳过程(参见 4.3 节)，且对于所有 i，有 $E(Y_i)=\nu$(或者，假定前 l 个观测值已被删除，我们处理的是 Y_{l+1}，Y_{l+2}，…。如果 ν 存在，l 足够大，则通常情况下 Y_{l+1}，Y_{l+2}，… 近似为协方差平稳过程)。假定我们进行一个长度为 m 的仿真运行，然后将所得到的观测值 Y_1，Y_2，…，Y_m 分成 n 批，每批长度都为 k(假设 $m=nk$)。因此，第 1 批由观测值 Y_1，Y_2，…，Y_k 组成，第 2 批由观测值 Y_{k+1}，Y_{k+2}，…，Y_{2k} 组成，以此类推。令 $\overline{Y}_j(k)$(其中 $j=1$，2，…，n)为第 j 批中 k 个观测值的

样本(或批)均值，并令 $\overline{\overline{Y}}(n,k)=\sum_{j=1}^{n}\overline{Y}_j(k)/n=\sum_{i=1}^{m}Y_i/m$ 为总样本均值，我们将用 $\overline{\overline{Y}}(n,k)$作为 ν 的点估计[$\overline{Y}_j(k)$对批均值法的作用与 X_j 在第 9.5.2 小节中的重复运行/删除法的作用最终是相同的]。

如果过程 Y_1，Y_2，…除了是协方差平稳外还满足其他一些条件，那么，对于固定的批数 n，Steriger 和 Wilson(2001)证明 $\overline{Y}_j(k)$的分布渐近(当 $k\to+\infty$)均值为 ν 的正态随机变量。因此，如果批长度 k 足够大，将所有 $\overline{Y}_j(k)$按服从均值 ν 的正态分布的独立同分布随机变量来对待是合理的。从而，将 $X_j=\overline{Y}_j(k)$代入到式(4.3)、式(4.4)与式(4.12)，就得到 ν 的点估计和近似百分之 $100(1-\alpha)$置信区间。

批均值法误差的主要来源在于批长度 k 选得过小，这导致 $\overline{Y}_j(k)$之间可能具有高相关性，以及 $S^2(n)/n$ 是 $\text{var}[\overline{X}(n)]=\text{var}[\overline{\overline{Y}}(n,k)]$的严重有偏的估计，参见第 4.4 节。特别地，如果 Y_i正相关(正如实际中往往是这种情况那样)，$\overline{Y}_j(k)$也一定会正相关，得到的方差估计偏低，且置信区间过小。从而，置信区间覆盖 ν 的概率就小于所要求的 $1-\alpha$。

文献中提出了一些变形批均值法。Meketon 和 Schmeiser(1984)介绍了*重叠批均值*(overlapping batch means，OBM)法，该方法中仍然使用$\overline{\overline{Y}}(n,k)$作为 ν 的点估计，但是 $\widehat{\text{var}}[\overline{\overline{Y}}(n,k)]$的表达式引进了长度为 k 的所有 $m-k+1$ 个均值。特别是，第 1 批由观测值 Y_1，Y_2，…，Y_k 组成，第 2 批由观测值 Y_{k+1}，Y_{k+2}，…，Y_{2k} 组成，以此类推。令 $\overline{Y}_j(k)$(其中 $j=1,2,\cdots,m-k+1$)为第 j 批中 k 个观测值的样本(或批)均值，则通常情况下，$\overline{Y}_j(k)$是高相关的。那么，$\widehat{\text{var}}[\overline{\overline{Y}}(n,k)]$基于 OBM 估计值可由下式得到：

$$\widehat{\text{var}}_{\text{O}}[\overline{\overline{Y}}(n,k)]=\frac{k\sum_{j=1}^{m-k+1}[\overline{Y}_j(k)-\overline{\overline{Y}}(n,k)]^2}{(m-k+1)(m-k)}$$

并且，ν 的近似百分之 $100(1-\alpha)$置信区间为：

$$\overline{\overline{Y}}(n,k)\pm t_{f,1-\alpha/2}\sqrt{\widehat{\text{var}}_{\text{O}}[\overline{\overline{Y}}(n,k)]}$$

其中，t 分布的自由度 f 在文献 Alexopoulos，Goldsman 和 Serfozo(2006)中进行了讨论。OBM 置信区间的试验结果可在 Sargent，Kang 和 Goldsman(1992)中找到。

Bitchak，Kelton 和 Pollak(1993)研究了*加权批均值法*的思想，该方法中，一批样本中第 i 个观测值的权重为 w_i，所有 w_i 之和等于 1。在普通的批均值法中，对于所有 i，有 $w_i=1/k$。Fox，Goldsman 和 Swain(1991)研究了*间隔批均值法*的思想，该方法中，用于实际分析的各个批之间插入一个大小为 s 的间隔，以减小 $\overline{Y}_j(k)$之间的相关性。

Tafazzoli 等(2011)提出了 N-Skark 法。利用式(4.13)给出的 Willink 置信区间说明了批均值的偏度，并且基于估计批均值间的滞后相关，对置信区间的半长进行了相关调整。

Argon 和 Andradóttir(2005)介绍了*重复批均值法*，该方法基于进行"少"量(r 次)长度为 m 的重复运行，然后将每个重复运行分成长度为 k 的 n 个批次($m=nk$)。然后将 r 次重复运行均值的样本均值用作 ν 的点估计，并用 rn 个批均值用于构建方差估计。对于该方法，当 $n=1$ 时，作为特定情形，包括重复运行法；对于该方法，当 $r=1$ 时，作为特定情形，还包括批均值法。其他一般性讨论批均值法的文章有：Alexopoulos 和 Goldsman(2004)、Alexopoulos，Goldsman 和 Serfozo(2006)、Damerdji(1994)、Fishman 和 Yarberry(1997)、Sargent，Kang 和 Goldsman(1992)、Schmeiser(1982)、Schmeiser 和 Song(1996)以及 Song 和 Schmeiser(1995)。基于批均值法的序贯法在本节的末尾进行讨论。

我们下面要讨论的自回归法和谱分析法两个方法不是力图得到独立性，而是利用所研究的随机过程的自相关结构的估计来得到样本均值的方差的估计，并最终构建 ν 的置信区

间。假设我们由单一的重复仿真所得到的观测值为 Y_1，Y_2，…，Y_m，并令 $\overline{Y}(m)=\sum_{i=1}^{m}Y_i/m$ 为 ν 的点估计。

自回归法。Fishman(1971，1973a，1978)研究出了自回归法，该方法假设 Y_1，Y_2，…为协方差平稳过程，$E(Y_i)=\nu$，并且可以采用 p 阶自回归模型来表示：

$$\sum_{j=0}^{p}b_j(Y_{i-j}-\nu)=\varepsilon_i \tag{9.7}$$

其中，$b_0=1$；$\{\varepsilon_i\}$为具有共同均值为0、方差为 σ_ε^2 且不相关随机变量序列。

对于已知的自回归阶数 p，且

$$\sum_{j=-\infty}^{+\infty}|C_j|<+\infty \tag{9.8}$$

可以证明，当 $m\to+\infty$时，$m\text{var}[\overline{Y}(m)]\to\sigma_\varepsilon^2/(\sum_{j=0}^{p}b_j)^2$。基于观测值 Y_1，Y_2，…，Y_m 的协方差 C_j 的估计，Fishman(1973a)给出了确定阶数 p 以及获得估计值$\hat{b}_j(j=1,2,\cdots,\hat{p})$和$\hat{\sigma}_\varepsilon^2$ 的方法，其中$\hat{p}$为估计的阶数。令 $\hat{b}=1+\sum_{j=1}^{\hat{p}}\hat{b}_j$，则对于大的 m，$\text{var}[\overline{Y}(m)]$的估计值以及 ν 的百分之 $100(1-\alpha)$置信区间为：

$$\widehat{\text{var}}[\overline{Y}(m)]=\frac{\hat{\sigma}_\varepsilon^2}{m(\hat{b})^2}$$

$$\overline{Y}(m)\pm t_{\hat{f},1-\alpha/2}\sqrt{\widehat{\text{var}}[\overline{Y}(m)]}$$

其中估计的自由度$\hat{f}$的表达式为：

$$\hat{f}=\frac{m\hat{b}}{2\sum_{j=0}(\hat{p}-2j)\hat{b}_j}$$

Yuan 和 Nelson(1994)给出了另一种估计自回归阶数 p 和自由度 f 的方法。对于 $\rho=0.9$ 的 $M/M/1$ 排队模型，该方法比 Fishman 的方法给出了更好的覆盖度。

在使用这些方法时主要的关注点是自回归模型是否对任意随机过程提供了一个好的表达式。Schriber 和 Andrews(1984)给出了一种广义的自回归法，该方法还考虑了滑动平均部分。

谱分析法也假设过程 Y_1，Y_2，…为协方差平稳且具有 $E(Y_i)=\nu$，但是不做如式(9.6)那样的进一步假设。在这种平稳假设下，可以证明：

$$\text{var}[\overline{Y}(m)]=\frac{C_0+2\sum_{j=1}^{m-1}(1-j/m)C_j}{m} \tag{9.9}$$

式(9.9)本质上与式(4.7)相同，谱分析法就是利用这个关系作为估计 $\text{var}[\overline{Y}(m)]$的出发点。该方法的名字是基于这样一个事实，倘若式(9.7)成立，则当 $m\to+\infty$时，我们有 $m\text{var}[\overline{Y}(m)]\to2\pi g(0)$，其中，$g(\tau)$称为过程在频率 τ 处的谱，并通过傅里叶(Fourier)变换 $g(\tau)=(2\pi)^{-1}\sum_{j=-\infty}^{+\infty}C_j\exp(-i\tau j)$ 来定义，其中，$|\tau|\leqslant\pi$ 且 $i=\sqrt{-1}$。因此，对于大的 m，$\text{var}[\overline{Y}(m)]\approx2\pi g(0)/m$，估计 $\text{var}[\overline{Y}(m)]$的问题可以看做估计在0频率处的谱。

将式(9.8)中的 C_j 替换为由 Y_1，Y_2，…，Y_m 和式(4.9)计算得到的$\hat{C}_j$，则可立即得到$\text{var}[\overline{Y}(m)]$的估计值。然而，对大的 m 和接近 m 的 j 来说，C_j 通常接近0，$\hat{C}_j$ 却会有大的方差，因为它将只基于少量的观测值。鉴于此，一些学者提出使用如下形式的估计：

$$\text{var}[\overline{Y}(m)]=\frac{\hat{C}_0+2\sum_{j=1}^{q-1}W_q(j)\,\hat{C}_j}{m}$$

其中，q(它决定估计值中$\hat{C}_j$ 的个数)需要事先指定；设计权重函数 $W_q(j)$是为了改进$\widehat{\text{var}}[\overline{Y}(m)]$的采样特性。那么，$\nu$ 的近似百分之 $100(1-\alpha)$置信区间为：

$$\overline{Y}(m)\pm t_{f,1-\alpha/2}\sqrt{\widehat{\text{var}}[\overline{Y}(m)]}$$

其中，f 取决于 m，q，以及权重函数的选择[参见 Fishman(1969，1973a)以及 Law 和 Kelton(1984)]。Welch(1987)讨论了批均值法、重叠批均值法以及谱分析法之间的关系。

谱分析法复杂，对部分分析人员来说需要相当好的知识基础。另外，没有确定的选择 q 值的方法。关于谱分析法的更多讨论可以在 Damerdji(1991)、Heigelberger 和 Welch (1981a，1981b，1983)、Lada 和 Wilson(2004)以及 Lada 等(2004)中找到。

再生法对仿真来说是一种完全不同的方法，因而导致不同的构建 ν 的置信区间方法。其思想是找到随机时间，在该时间点过程在概率意义下“重新开始”即再生，并利用这些再生点来获得独立的随机变量，对这些随机变量能应用经典的统计分析方法以构建 ν 的点估计和区间估计。该方法是由 Crane 和 Iglehart(1974a，1974b，1975)以及 Fishman (1973b，1974)并行开发出来的，我们遵循前者的表述。

假设对于输出过程 Y_1，Y_2，…，存在随机索引序列 $1\leqslant B_1<B_2<\cdots$，称为再生点，在这些点处该过程在概率意义下重新开始，也就是说，对于每个 $j=1$，2，…，过程 $\{Y_{B_j+i-1}$，$i=1$，2，…$\}$的分布都相同，并且从每个 B_j 开始的过程假设与 B_j 之前的过程独立。两个相邻 B_j 之间的那部分过程分称为再生周期，并且能够证明相邻周期是互相独立且同分布重复的。特别地，定义在相邻周期上的可比随机变量是独立同分布的。对于 $j=1$，2，…，令 $N_{j+1}=B_{j+1}-B_j$，并假设 $E(N_j)<+\infty$。如果 $Z_j=\sum_{j=B_j}^{B_{j+1}-1}Y_j$，则随机向量 $\boldsymbol{U}_j=(Z_j，N_j)^{\mathrm{T}}$(其中，$\mathbf{A}^{\mathrm{T}}$是向量 $\mathbf{A}$ 的转置)是独立同分布的，且倘若 $E(|Z_j|)<+\infty$，则稳态均值 ν 为(参见习题 9.21)：

$$\nu=\frac{E(Z)}{E(N)}$$

例 9.32 考虑一个单服务台排队系统的延误时间的输出过程 D_1，D_2，…具有 IID 到达间隔时间，IID 服务时间，顾客服务为先到先服务方式，并且 $\rho<1$。到达时发现系统完全空的那些顾客的索引号为再生点(参见图 9.17)。令 N_j 表示第 j 个周期中被服务的顾客总数，$Z_j=\sum_{i=B_j}^{B_{j+1}-1}D_i$ 表示第 j 个周期中被服务的所有顾客的总延误时间，则稳态平均延误时间为 $d=E(Z)/E(N)$。

注意，一般说来，到达时发现有 $l(l\geqslant 1$，且固定)个顾客的顾客索引号不是过程 D_1，D_2，…的再生点，原因是服务中的顾客的剩余服务时间的分布与到达时发现有 l 个顾客的后续顾客的剩余服务时间分布是不同的。然而，如果服务时间是指数随机变量，则这些索引号是再生点，因为指数分布具有无记忆特性(参见习题 4.26 和习题 9.22)。 ◀

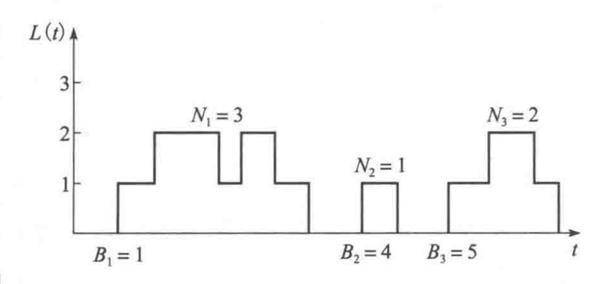


图 9.17 单服务台排队系统的系统中顾客数过程 $\{L(t)，t\geqslant 0\}$的一个实现

我们现在讨论如何使用再生法得到 ν 的点估计和置信区间。假定我们对过程

Y_1，Y_2，…进行了正好 n'个再生周期的仿真，则可得到如下数据：

$$Z_1, Z_2, \cdots, Z_{n'}$$
$$N_1, N_2, \cdots, N_{n'}$$

两个序列都由 IID 随机变量组成。然而，通常情况下，Z_j 与 N_j 不独立。那么，ν 的点估计为：

$$\hat{\nu}(n') = \frac{\overline{Z}(n')}{\overline{N}(n')}$$

虽然 $\overline{Z}(n')$和 $\overline{N}(n')$分别是 $E(Z)$和 $E(N)$的无偏估计，但是$\hat{\nu}(n')$并非 ν 的无偏估计(参见附录 9A)。然而，可以肯定的是，$\hat{\nu}(n')$为 ν 的强一致估计，即当 $n'\to+\infty$时，$\hat{\nu}(n')\to\nu$(以概率 1)，参见习题 9.21。

令向量 $\boldsymbol{U}_j=(Z_j,\ N_j)^{\mathrm{T}}$ 的协方差矩阵为：

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$$

例如，$\sigma_{12}=E\{[Z_j-E(Z_j)][N_j-E(N_j)]\}$，并令 $V_j=Z_j-\nu N_j$，则这些 V_j 为独立同分布的随机变量，且均值为 0，方差 $\sigma_V^2=\sigma_{11}-2\nu\sigma_{12}+\nu^2\sigma_{22}$(参见习题 4.13)。那么，如果 $0<\sigma_V^2<+\infty$，则由经典的中心极限定理(参见 4.5 节的定理 4.1)可以得到：

$$\frac{\overline{V}(n')}{\sqrt{\sigma_V^2/n'}} \xrightarrow{\mathscr{D}} N(0,1), \quad n'\to+\infty \tag{9.10}$$

其中，$\xrightarrow{\mathscr{D}}$表示依分布收敛。令

$$\hat{\boldsymbol{\Sigma}}(n') = \begin{bmatrix} \hat{\sigma}_{11}(n') & \hat{\sigma}_{12}(n') \\ \hat{\sigma}_{21}(n') & \hat{\sigma}_{22}(n') \end{bmatrix} = \frac{\sum_{j=1}^{n'}[\boldsymbol{U}_j-\overline{\boldsymbol{U}}(n')][\boldsymbol{U}_j-\overline{\boldsymbol{U}}(n')]^{\mathrm{T}}}{n'-1}$$

为估计的协方差矩阵，并令

$$\hat{\sigma}_V^2(n') = \hat{\sigma}_{11}(n') - 2\,\hat{\nu}(n')\,\hat{\sigma}_{12}(n') + [\hat{\nu}(n')]^2\,\hat{\sigma}_{22}(n')$$

为基于 n'个再生周期的 σ_V^2 的估计值。可以证明当 $n'\to+\infty$时$\hat{\sigma}_V^2(n')\to\sigma_V^2$(以概率 1)。从而，我们可以用$\hat{\sigma}_V^2(n')$代替式(9.9)中的 σ_V^2[参见 Chung(1974，第 93 页)]，并将比值除以 $\overline{N}(n')$，可以得到：

$$\frac{\hat{\nu}(n')-\nu}{\sqrt{\hat{\sigma}_V^2/\{n'[\overline{N}(n')]^2\}}} \xrightarrow{\mathscr{D}} N(0,1), n'\to+\infty$$

因此，如果周期数 n'足够大，ν 的近似(依照覆盖度)百分之 $100(1-\alpha)$置信区间为：

$$\hat{\nu}(n') \pm \frac{z_{1-\alpha/2}\sqrt{\hat{\sigma}_V^2(n')/n'}}{\overline{N}(n')} \tag{9.11}$$

我们称这种构造 ν 的置信区间的再生法为经典法(C)。另外一种构造 ν 的置信区间的再生法称为对折(jackknife)法(J)，参见附录 9A。

在实际中使用再生法的困难在于，实际问题的仿真中可能不存在再生点，或者(即使存在)有可能周期过大使得只能仿真少数周期[在这种情形下由式(9.11)得到的置信区间失效]。例如，假定人们希望通过仿真估计由 k 个串行排队系统组成的网络的队列中稳态顾客总延误时间[离开排队系统的 i(其中 $i=1,\ 2,\ \cdots,\ k-1$)顾客前进到第 $i+1$ 的位置]，则过程 D_1，D_2，…(其中 D_i 表示第 i 个到达顾客的总延误时间)的再生点是那些到达第一个排队系统并发现整个网络系统为空的顾客的索引号。如果组成该网络的排队系统的利用率高，那么典型情况就是，该网络的再生点将会非常少，而且彼此相隔很远。关于再生法的更完整的讨论可以在 Crane 和 Lemoine(1997)、Henderson 和 Glynn(2001)以及 Schedler(1993)中找到。

标准时间序列法[参见 Schruben(1983a)]。假设过程 Y_1，Y_2，…为严格平稳的，对于

所有 i，有 $E(Y_i)=\nu$，而且还是 Φ-混合的。严格平稳的含义是，对于所有时间索引号 i_1，i_2，…，i_n，Y_{i_1+j}，Y_{i_2+j}，…，Y_{i_n+j} 的联合分布是与 j 独立的(那么，如果 ν 存在，通常情况下，如果 l 足够大，Y_{l+1}，Y_{l+2}，…应该是近似严格平稳的)。粗略地说，如果随着 j 的增大，Y_i 和 Y_{j+i} 变得本质独立，则 Y_1，Y_2，…为 Φ-混合的[更精确的定义，请参见 Billingsley(1999)]。假定我们进行了一次长度为 m 的仿真，并将 Y_1，Y_2，…，Y_m 分成大小为 k 的 n 批(其中 $m=nk$)。令 $\overline{Y}_j(k)$ 表示第 j 批中 k 个观测值的样本均值。总样本均值 $\overline{Y}(m)$ 为 ν 的点估计。更进一步，如果 m 很大，则 $\overline{Y}(m)$ 将近似为均值为 ν、方差为 τ^2/m 的正态分布，其中，

$$\tau^2 = \lim_{m\to+\infty} m\mathrm{var}[\overline{Y}(m)]$$

并称为方差参数。令

$$A = \left(\frac{12}{k^3-k}\right)\sum_{j=1}^{n}\left\{\sum_{s=1}^{k}\sum_{i=1}^{s}[\overline{Y}_j(k)-Y_{i+(j-1)k}]\right\}^2$$

对于固定的批数 n，A 渐近(当 $k\to+\infty$)分布为自由度为 n 的 χ^2 随机变量的 τ^2 倍，并且与 $\overline{Y}(m)$ 渐近独立。因此，当 k 很大时，我们可将

$$\frac{[\overline{Y}(m)-\nu]/\sqrt{\tau^2/m}}{\sqrt{(A/\tau^2)/n}} = \frac{\overline{Y}(m)-\nu}{\sqrt{A/(mn)}}$$

作为具有自由度为 n 的 t 分布来对待，并且 ν 的近似百分之 $100(1-\alpha)$ 置信区间为：

$$\overline{Y}(m)\pm t_{n,1-\alpha/2}\sqrt{A/(mn)}$$

标准时间序列法误差的主要来源是批大小 k 过小[细节参见 Schruben(1983a)]。应该指出的是，该方法的基础理论与第 9.5.1 小节讨论的关于初始偏差的 Schruben 测试是一样的。关于标准时间序列法的其他参考文献包括其他的置信区间公式，有 Glynn 和 Iglehart (1990)、Goldsman，Meketon 和 Schruben(1990)、Goldsman 和 Schruben(1984，1990)，以及 Sargent，Kang 和 Goldsman(1992)。

鉴于本节阐述的 5 种固定样本长度置信区间方法所依赖的假设条件在实际仿真中得不到严格满足，人们关心这些方法在实际中的表现如何。我们首先给出 $\rho=0.8$ $\left(\lambda=1\text{ 且 }\omega=\frac{5}{4}\right)$ 的 $M/M/1$ 排队系统的 400 次独立仿真实验的结果，其中每次试验的目标是使用这 5 种方法来构建稳态平均延误时间 $d=3.2$ 的 90% 置信区间。对于批均值法(B)、自回归法(A)、谱分析法(SA)，以及标准时间序列法(STS)，由于不确知如何选择样本总量 m，我们任意选取 $m=320$、640、1 280 和 2 560。对于再生法(R)，可以证明，对于 $\rho=0.8$ 的 $M/M/1$ 排队系统，$E(N)=1/(1-\rho)=5$(参见习题 9.25)。因此，我们选择再生周期数 $n'=64$、128、256 和 512，以使得所有方法平均说来所使用的观测值个数相同，即 $m=n'E(N)$。进一步，我们考虑了经典的和对折的两种再生置信区间。对于批均值法和标准时间序列法，我们选择批数 $n=5$、10 和 20。谱分析法的自由度 f 的选择满足 $f+1=n$，其中 f 与方差表达式 $q=1.33m/f$ 中协方差估计 q 的个数有关[细节请参考文献 Law 和 Kelton(1984)]。表 9.9 给出了上面讨论的 48 种情况的每一种所得到的 400 个置信区间覆盖 d 的比例[除了标准时间序列法的数据由乔治亚理工的 David Goldsman 提供之外，所有结果均来自 Law 和 Kelton(1984)]。例如，批均值法在 $m=320$ 且 $n=5$ 的情况下(即每个置信区间基于 5 批大小为 64)，400 个置信区间中覆盖 d 的比例为 69%，明显地小于所期望的 90%(注意，对固定的 m，批均值法估计的覆盖度随着 n 的增大而减小。这是因为，随着 n 的增大，批均值变得更加相关，这导致样本均值的方差的估计更偏)。

表 9.9 基于 400 次实验的估计的覆盖度，$\rho=0.8$ 的 $M/M/1$ 排队系统

	B			STS			SA			A	R	
	n			n			$f+1$				方法	
$m(n')$	5	10	20	5	10	20	5	10	20		C	J
320(64)	0.690	0.598	0.490	0.520	0.340	0.208	0.713	0.625	0.538	0.688	0.560	0.670
640(128)	0.723	0.708	0.588	0.628	0.485	0.318	0.760	0.735	0.645	0.723	0.683	0.728
1280(256)	0.780	0.740	0.705	0.730	0.645	0.485	0.783	0.770	0.745	0.753	0.705	0.748
2560(512)	0.798	0.803	0.753	0.798	0.725	0.598	0.833	0.808	0.773	0.755	0.745	0.763

接下来，我们给出有 35 个终端的分时计算机模型的 200 次独立仿真实验结果[见 Law 和 Kelton(1984)]，这在 2.5 节讨论过。我们的目标是构建稳态平均响应时间 $r=8.25$ 的 90%置信区间[参见 Adiri 和 Avi-Itzhak(1969)]。我们按照上面的方法选择 m 和 n 的值，因为计算机模型的 $E(N)\approx 32$，所以我们取 $n'=10$、20、40 和 80。表 9.10 给出了 36 种情况的每一种的 200 个置信区间覆盖 r 的比例(标准时间序列法的结果不存在)。虽然计算机模型从物理上看要比 $M/M/1$ 排队系统复杂得多，但是从表 9.10 可以看出，对于 $n=5$ 的批均值方法，即使 m 值小到 640，得到估计的覆盖度也非常接近 0.90。因此，$\rho=0.8$ 的 $M/M/1$ 排队系统，尽管结构非常简单，但在统计上困难得多。这两个例子说明，不能根据模型结构的复杂程度来推断其输出数据的统计行为特性。

表 9.10 基于 200 次实验的估计的覆盖度，分时计算机模型

	B			SA			A	R	
	n			$f+1$				方法	
$m(n')$	5	10	20	5	10	20		C	J
320(10)	0.860	0.780	0.670	0.880	0.815	0.720	0.680	0.545	0.725
640(20)	0.890	0.855	0.790	0.870	0.870	0.820	0.805	0.730	0.830
1280(40)	0.910	0.885	0.880	0.910	0.910	0.905	0.890	0.830	0.865
2560(80)	0.905	0.875	0.895	0.910	0.885	0.900	0.885	0.870	0.915

根据表 9.9 和表 9.10 的试验结果以及文献 Law(1977)，Law 和 Kelton(1984)，Sargent，Kang 和 Goldsman(1992)，关于固定样本长度法，我们可以得到如下结论。

(1) 如果样本总数 m(或者 n')选得过小，所有现有的固定样本长度法(包括重复运行/删除法)的实际覆盖度可能远远小于所要求的。这一点并不奇怪，因为稳态参数的定义为仿真长度(即观测值的总个数)趋于无穷时的极限。

(2) m(或者 n')值的"合适"选取与模型本身有很大的关系，不能随意确定。对于 $n=5$ 的批均值法，$m=640$ 对于计算机模型得到了很好的结果；然而对于 $M/M/1$ 排队系统，即使 m 大到 2 560，我们依然不能得到满意的结果。

(3) 当 m 固定时，批均值法、标准时间序列法，以及谱分析法在 n 和 f 较小的情况下会达到最好的覆盖度。

序贯法

我们现在讨论序贯地确定单一仿真运行所需长度的方法以构建稳态均值 ν 一个可接受的置信区间。从上面给出的固定样本长度法的结果可以明显看出需要这种序贯法。特别是，如果固定运行长度对被仿真的系统来说太小的话，在仿真开始前，对于固定运行长度的方法，一般来说，没有一种方法可保证得到的置信区间以所要求的概率 $1-\alpha$ 覆盖 ν。

除了覆盖度问题以外，分析人员可能希望确定运行长度足够大以得到具有指定的绝对误差 β 或者相对误差 γ 的 ν 的估计(参见第 9.4.1 小节)。对于一个给定的仿真问题，甚至

不可能事先知道为满足这些目标所需的仿真运行长度的数量级，因此某种迭代地增加运行长度的方法似乎是合理的。

Law 和 Kelton(1982)，以及 Law(1983)对当时可用的序贯法进行了综述，并发现，如果指定的绝对误差或相对误差足够小，根据所达到的覆盖度有三种方法比较好。特别地，Fishman(1997)开发了一种基于再生法和绝对误差停止规则的方法。Law 和 Kelton 发现，对于测试的 10 个随机模型，如果 $\beta=0.075\nu$，则有 9 个达到了可接受的覆盖度。Fishman 方法的缺点在于它是基于再生法的，我们觉得这个缺点限制了该方法在实际问题中的应用。并且，在实际问题中，确定一个合适的 β 值有可能十分困难，当然是因为 ν 通常是未知的。

Law 和 Carson(1979)开发了一种基于批均值法和相对误差停止规则的方法。对于固定的批数 $n=40$，逐渐增加批大小 k 直到所得的批均值近似不相关，并且相应的置信区间满足指定的相对误差为止。但是，在特定迭代中，实际上利用 400 个批均值(每个批均值基于 $k/10$ 个观测值)来确定相应的 40 个批均值(每个基于 k 个观测值)是否不相关，这个方案的必要性在于，相关性估计一般是有偏的，并且小的 n 有大的方差。Law 和 Carson (L&C)方法没有测试这些批均值是否服从正态分布，但是因为 $n=40$ 并且每一个批均值均为大量独立观察值的平均值，一般情况下，这可能不是一个主要问题。Law 和 Carson 将他们的方法应用到了 14 个可以通过解析方法计算 ν 值的随机模型中。对于每个模型，他们试图构建相对误差 $\gamma=0.075$ 的 90%置信区间，并且进行了 100 次独立的实验(通常情况下，我们认为如果有人要使用序贯法，则他们将选择 $\gamma\leqslant 0.1$。否则，他们会用固定样本大小法)。注意，L&C 方法没有明确地解决启动问题(例如，使用无预热周期)。

在表 9.11 中，我们分别给出了 5 个测试模型采用 Law 和 Carson 方法(L&C)所得到的 100 个置信区间(包含 ν)、真实覆盖度 p 的 90%置信区间，终端上的平均运行长度(即样本大小)。真实覆盖度 p 的 90%置信区间可由 $\hat{p}\pm z_{0.95}\sqrt{\hat{p}(1-\hat{p})/100}$ 计算得来(参见第 9.4.2 小节)，表 9.11 的前 4 行的结果分别对应下列 4 个模型在队列中的延误时间过程：$M/M/1$ 排队模型，$M/M/1$ 后进先出排队模型，$M/H_2/1$ 排队模型[具有 cv＝2 的超指数分布服务时间，准确定义参见 Law(1974)]，以及 $M/M/1/M/1$ 排队模型(即两个串行的 $M/M/1$ 排队模型)。对于每个模型，$\rho=0.8$。表 9.11 的第 5 行对应计算机系统的简单模型的响应时间过程，Law 和 Carson 称该模型为中央服务器模型 3。表 9.11 的最后一行给出了对所有方法进行评估的独立试验的次数。

表 9.11 $\gamma=0.075$ 时构建名义上的 90%置信区间的估计覆盖度、真实覆盖度的 90%置信区间，以及平均样本长度，批均值法

模型	L&C	ASAP3	SBatch	Skart
$M/M/1$，$\rho=0.8$	0.87±0.06	0.868±0.028	0.903±0.024	0.912±0.014
	75 648	72 060	89 434	82 508
$M/M/1$ LIFO，$\rho=0.8$	0.84±0.06	0.875±0.027	0.888±0.016	0.916±0.014
	74 624	68 325	97 172	81 441
$M/H_2/1$，$\rho=0.8$	0.90±0.05	0.900±0.025	0.896±0.025	0.913±0.015
	229 632	228 482	254 400	255 363
$M/M/1/M/1$，$\rho=0.8$	0.87±0.06	0.913±0.023	0.931±0.013	0.909±0.016
	49 920	58 844	55 398	58 573
中央服务器模型 3	0.88±0.05	0.870±0.026	0.921±0.014	0.873±0.017
	3 740	18 447	85 842	12 231
试验次数	100	400	1 000	1 000

Heidelberger 和 Welch(1983)开发了一种基于谱分析法和相对误差停止规则的方法(H&W)，利用回归技术估计频率为零处的频谱。他们的方法要求用户指定最大样本大小，那么生成的某些置信区间可能会不满足相对误差的要求。对于计算机系统的两个模型的时间响应过程，他们试图构建 ν 的 90%置信区间，并完成了 50 次独立试验[参见 Heidelberger 和 Welch(1981b)]。令$\hat{p}$为满足覆盖ν的精度要求的比率。对于 $\gamma=0.05$，他们取两个模型的$\hat{p}$分别等于 0.08 和 0.84。表 9.12 给出了 H&W 方法的其他的实证结果。

Steiger 和 Wilson(2005)提出了自动仿真分析法的修改版本[Steiger 和 Wilson(2002)]，称为 ASAP3，该方法是基于批均值方法的。其原理如下：逐步增大批数直到 4 个相邻批均值构成的间隔组通过多变量正态性检验(参见第 6.10.1 小节)，其中，每组之前的间隔中同样包含四个相邻的批均值。随后，当跳过作为预热周期的第一个间隔后，ASAP3 满足了一个无间隔批均值 AR(1)过程。如果必要的话，批量大小将会进一步增加，直到AR(1)模型中的自回归参数 ϕ 不超过 0.8。接下来，ASAP3 利用一个基于 AR(1)参数的修正的 t 置信区间估计来说明保留批均值中的相关性。注意，不同于 L&C 法，ASAP3 没有试图获得不相关的批均值。对于每一个模型，他们尝试构建 ν 的相对误差为$\gamma=0.075$ 的 90%置信区间，并完成了 400 次独立试验，其结果如表 9.11 所示。Tafazzoli 等(2011)给出了 ASAP3 其他的性能结果。

Lada 等(2008)介绍了构建稳态均值的置信区间的 SBatch(间隔批均值)法。SBatch 法利用随机检验和正态检验来反复确定每个批量的间隔大小 s 和批量大小 k，则生成的间隔批均值为近似服从独立同分布的正态随机变量。为了检查批均值间的任何剩余相关性，SBatch 法检验了间隔批均值的滞后相关性以确保其值不超过 0.8。每当相关性检验失败时，则增加批量大小，其他观察值来自于仿真，计算新的间隔批均值集合并反复对其进行相关性检验。

一旦通过了相关性检验，SBatch 法利用现有的间隔批均值集合构建了 ν 的相关性调整的置信区间：置信区间的中心为除第一个间隔(预热周期)之外的所有观察值的批均值，置信区间的半长使用间隔批均值的样本方差以及基于间隔批均值间的估计滞后相关性的相关性调整[见式(4.9)]。他们对每一个模型都试图构建 ν 的相关误差 $\gamma=0.075$ 的 90%的置信区间，并且完成了 1 000 次独立试验，其结果如表 9.11 所示(非常感谢 Dr. Emily Lada，Dr. Ali Tafazzoli 以及 James Wilson 教授为表 9.11 中的结果所做的贡献)。Tafazzoli 等(2011)给出了 SBatch 法的其他结果。

Tafazzoli 和 Wilson(2011)研发了一种命名为 Skart(偏度-自回归-调整的 Student$t$ 分析)的方法[参见 Tafazzoli 等(2011)]。Skart 法通过渐进增加批量大小和批间间隔大小，成功地对间隔批均值进行了随机检验，从而解决了启动问题。在批量大小为 k、间隔大小为 dk(d 为非负整数)的条件下，最终通过了检验时，预热周期为 $l=dk$。

对于超过预热周期 l 的数据，Skart 法计算并使用批量大小为 k 的 n' 无间隔批均值。通常情况下，因为批均值会偏斜，Skart 法利用式(4.13)[Willink(2005)]构建了稳态均值 ν 的置信区间，其中批均值为 X_i。非对称置信区间的“中心”为批均值的样本均值，半长使用间隔批均值的样本方差，以及基于间隔批均值间的估计滞后相关性的相关性调整。他们对每一个模型都试图构建 ν 的相关误差 $\gamma=0.075$ 的 90%的置信区间，并且完成了 1 000 次独立试验，其结果如表 9.11 所示。

Lada 和 Wilson(2006a)开发了一种使用相对误差停止规则的基于小波变换的谱分析法。Lada 和 Wilson(2006)在之前讨论过的 5 个模型中测试了 WASSP 法以及 H&W 法。他们完成了 400 次独立试验并并力图构建相对误差 $\gamma=0.075$ 的 90%置信区间，其结果如表 9.12 所示。根据表 9.12 以及 Lada 等(2007)中的覆盖度结果，我们认为较平均样本长度更为重要，WASSP 法更优于 H&W 法。Tafazzoli 等(2011)给出了 WASSP 法的其他结果。

表 9.12 $\gamma=0.075$ 时构建名义上的90%置信区间的估计覆盖度、真实覆盖度的90%置信区间，以及平均样本长度，频谱法

模型	H&W	WASSP
$M/M/1$，$\rho=0.8$	0.790 ± 0.034	0.885 ± 0.026
	77 971	117 540
$M/M/1$ LIFO，$\rho=0.8$	0.795 ± 0.033	0.902 ± 0.024
	80 098	152 355
$M/H_2/1$，$\rho=0.8$	0.780 ± 0.034	0.910 ± 0.024
	233 430	330 580
$M/M/1/M/1$，$\rho=0.8$	0.803 ± 0.033	0.890 ± 0.026
	52 700	82 680
中央服务器模型 3	0.880 ± 0.027	0.930 ± 0.021
	12 562	79 188
试验次数	400	400

但是，我们确实认为 ASAP3 法、SBatch 法和 SKart 法要优于 WASSP 法，因为通常情况下后者所需的平均样本长度要比前者大得多。同时，WASSP 法也更为复杂，例如，相较于 SKart 法需要 13 步，该方法需要 21 步(如果对于这些方法中的其中一种方法，一个特殊的步骤需要 p 个子步骤，则我们认为整体步骤实际上是由 p 个步骤组成的)。根据表9.11所示的结果，以及 Tafazzoli 等(2011)中的表 1 和表 6 的结果，在 ASAP3 法、SBatch 法和 SKart 法中，以覆盖度和平均样本长度作为比较标准，我们认为 SKart 法提供的整体结果最好。

但是，我们认为 L&C 法同样值得思考，因为在 14 个随机模型中提供了很好的性能，尽管每个模型仅基于 100 次试验(相较于 SKart 的 1 000 次试验)。L&C 法同样也是在之前所讨论的所有序贯法中最简单的一种方法，仅需要 5 步，如果一种方法针对一个特别应用而需要重新编程那么这个方法是非常重要的。最后，Chen 和 Kelton(2009)基于批均值提出了另一种的序贯法，但是仅给出了有限的和无法评比的试验结果。

如果人们要构建的稳态均值 ν 的置信区间很可能具有接近 $1-\alpha$ 的覆盖度并需要“合理的”样本长度，则可以考虑使用 ASAP3 和 L&C 方法，取相对误差 $\gamma=0.075$ 或更小。这两种方法至少在 6 个模型上进行了测试，并且得到的估计覆盖度与平均样本长度的结果都很好。然而，读者需要意识到，相对于 9.5.2 节的重复运行/删除法，这两种方法的理解和实现要略微复杂一些。还有，这两种方法需要较大的样本长度，而对多种性能度量的一般情形来说，这两种方法不易产生[参见 9.7 节以及 Tafazzoli 等(2011)]。

需要注意的是，Glynn 和 Whitt(1992b)给出了序贯程序法渐近有效，即当运行长度 m 趋于无穷时产生覆盖度 $1-\alpha$ 的充分条件。

9.5.4 估计性能的其他度量

正如我们在第 9.4.2 小节所看到的，均值并不总为我们提供合适的系统性能度量。因此，我们考虑除均值 $\nu=E(Y)$以外的稳态参数 φ 的估计。

假定希望估计稳态概率 $p=P(Y\in B)$，其中 B 为一个实数集。举例来说，对于通信网络；我们希望得到报文端到端的延迟时间小于或等于 5 秒的稳态概率($B=\{$所有小于或等于 5 的实数$\}$)。事实证明，正如我们将要看到的，估计概率 p 只是估计均值 ν 的一种特殊情况。令稳态随机变量 Z 定义为：

$$Z=\begin{cases}1, & Y\in B\\ 0, & \text{其他}\end{cases}$$

则

$$P(Y \in B) = P(Z=1) = 1 \times P(Z=1) + 0 \times P(Z=0) = E(Z)$$

所以，估计 p 等价于估计稳态均值 $E(Z)$，后者在 9.5.2 节和 9.5.3 节中已经讨论过了。特别地，对于 $i=1$，2，…，令

$$Z_i = \begin{cases} 1, & Y_i \in B \\ 0, & \text{其他} \end{cases}$$

其中，Y_1，Y_2，…为所关心的原随机过程。

那么，例如，对于输出过程 Z_1，Z_2，…，可应用重复运行/删除法得到 $E(Z)=p$ 的点估计和置信区间。注意，(二进制)过程 Z_1，Z_2，…的预热周期与原过程 Y_1，Y_2，…的也许不同。

人们特别关心的另一个稳态分布的参数是 q-分位数 y_q，其在 6.4.3 节已经定义过。因此，y_q 表示满足 $P(Y \leqslant y_q)=q$ 的 y 值，其中 Y 为稳态随机变量。例如，在上面讨论的通信网络中，可能需要估计稳态端到端延迟时间分布的 0.9-分位数。不论在概念上还是在计算复杂度(根据为获得指定精度所需的观测值数量)方面，估计分位数较估计稳态均值更为困难。而且，大多数估计分位数的方法都基于统计的方法，且需要对观测值进行存储和排序。

有若干基于批均值(或扩展)法、谱分析法以及再生法的估计分位数的方法已经提出[见文献 Law(1983)以及 Heidelberger 和 Lewis(1984)]。这些方法的一个缺陷是，它们都是基于固定的样本长度，而样本长度的选择必然有某种程度的随意性。如果该样本长度选得过小，所得置信区间的覆盖度就多少有可能小于期望值。

Raatikainen(1990)提出了一种基于 Jain 和 Chlamtac(1985)的 P^2 算法的估计分位数的方法，该方法无需对观测值进行存储和排序。它是一种基于谱分析法和相对误差停止规则的序贯法。Raatikainen 在计算机系统的多个随机模型上对他的方法进行了测试，结果得到了很好的覆盖度。然而，这个方法难于实现。

Chen 和 Kelton(2005)提出了两种序贯法——窗口放大法(zoom in，ZI)和准独立法(quasi-independent，QI)，以构建分位数的置信区间。他们对 $M/M/1$ 和 $M/M/2$ 排队模型的在队列中的延误时间过程进行了方法测试，每种情况进行 100 次独立试验。举例来说，假定目标是构建 $\rho=0.9$ 的 $M/M/1$ 排队模型的 0.9-分位数的 95%置信区间。ZI 法的估计覆盖度为 1.00，终止时平均样本长度大约为 12 464 000。另一方面，QI 法的估计覆盖度为 0.95，终止时平均样本长度大约为 14 793 000。

Alexopoulos 等(2012)研究了基于非重叠批次开发贯序程序对分位数构建点估计和置信区间的可行性(批分位数渐进独立)。在所需样本大小方面，他们的结果是令人鼓舞的。

9.6 稳态周期参数的统计分析

假定输出过程 Y_1，Y_2，…不存在稳态分布。另一方面，假设有一个合适的周期定义使得过程 Y_1^C，Y_2^C，…具有稳态分布 F^C，其中，Y_i^C 为定义在第 i 个周期上的随机变量(参见 9.3 节)。如果 $Y^C \sim F^C$，那么我们关心的是估计 Y^C 的某些特征值，如均值 $\nu^C=E(Y^C)$ 或者概率 $P(Y^C \leqslant y)$。显然，估计稳态周期参数只是估计稳态参数的一种特殊情况，所以 9.5 节中的所有方法都适用于周期随机变量 Y_i^C 而不是原来的 Y_i。例如，我们可以使用韦尔奇方法来确定预热周期，而后应用重复运行/删除法得到 ν^C 的点估计和置信区间。

例 9.33 再次考虑例 9.25 中的小型工厂，但是假定在每个 8 小时班次的第 4 个小时开始一个长度为半小时的午休。在午休时间里，停止检测过程，但是未完成的工件继续到达并由无人机床进行加工。如果 N_i 表示第 i 个小时的产量，则过程 N_1，N_2，…不存在稳态分布(见例 9.11)。然而，我们可以期望它是周期性的，周期长度为 8 小时。为了证实这一点，我们进行了 $n=10$ 次长度 $m=160$ 小时(20 个班次)的重复运行。图 9.18 给出了平均的过程 $\overline{N}_i$(其中 $i=1$，2，…，160)的图形，从图中可看到，过程 N_1，N_2，…确实呈现出一个长度为 8 小时的周期。

令 N_i^C 表示第 i 个 8 小时周期内平均产量，并假设 N_1^C，N_2^C，…具有稳态分布。假定希望使用重复运行/删除法得到一个班次内稳态期望平均产量 $\nu^C=E(N^C)$ 的点估计和 99% 置信区间。令 N_{ji}^C 为第 j 次可用的重复运行的第 i 个周期的平均产量($j=1$，2，…，10；$i=1$，2，…，20)，并对于 $i=1$，2，…，20，令 $\overline{N}_i^C$ 为相应的平均过程（即 $\overline{N}_i^C=\sum_{j=1}^{10}N_{ji}^C/10$），图 9.19 画出了它的图形。从图中可得出结论，需要进一步对其进行平滑。因此，对于 $w=3$ 和 $w=6$ 个班次，我们在图 9.20a 和图 9.20b 中画出了移动平均 $\overline{N}_i^C(w)$(来自韦尔奇法)的图形。根据 $w=6$ 的图形(它更为平缓)，我们选择预热周期 $l=5$ 个班次或 40 小时(请将这里的 l 与例 9.25 中所得到的 l 进行比较)。

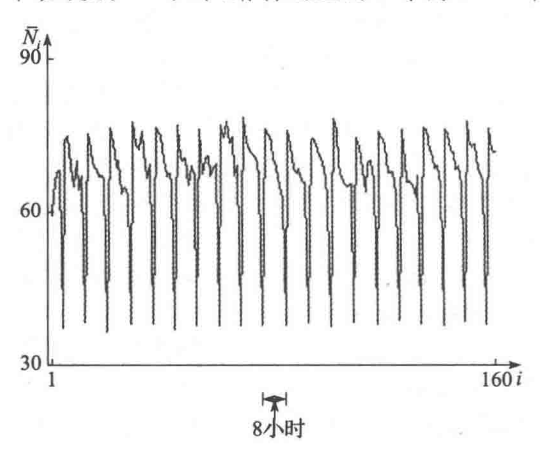


图 9.18 每小时生产量的平均过程，有午休的小型工厂

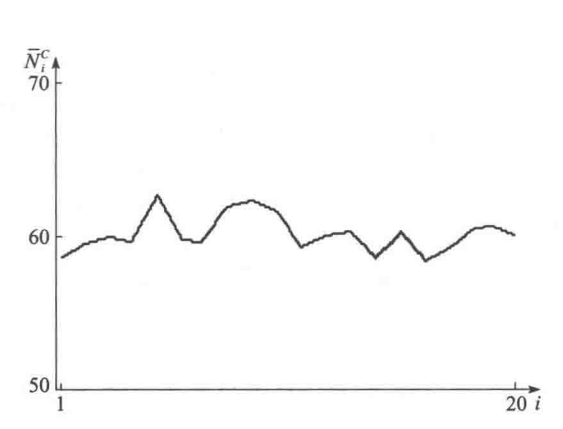


图 9.19 一个班次内的平均每小时产量的平均过程，有午休的小型工厂

令

$$X_j^C=\frac{\sum_{i=6}^{20}N_{ji}^C}{15},\quad j=1,2,\cdots,10$$

则，ν^C 的点估计和 99% 置信区间分别为：

$$\hat{\nu}^C=\overline{X}^C(10)=60.24$$

以及

$$\overline{X}^C(10)\pm t_{9,0.995}\sqrt{\frac{0.79}{10}}=60.24\pm 0.91$$

这也包含 60(见习题 9.27)。

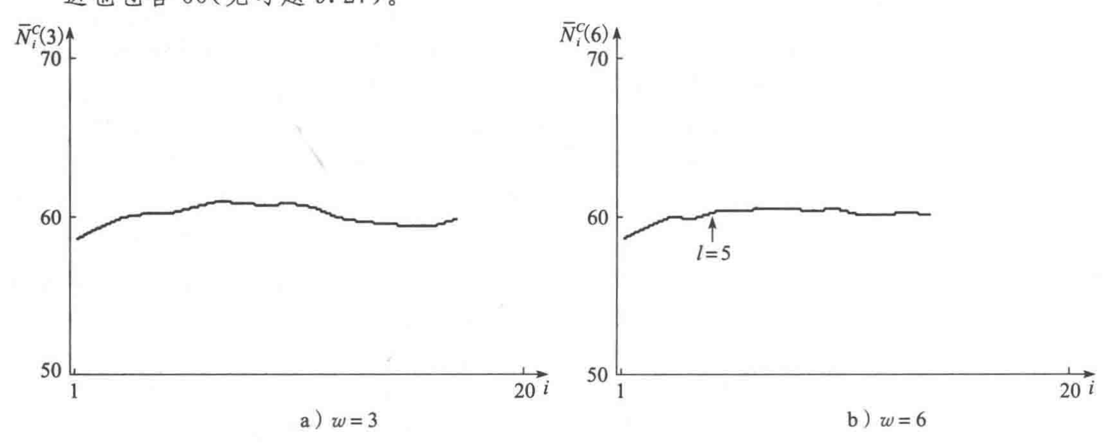


图 9.20 一个班次内的平均每小时产量的移动平均，有午休的小型工厂 ◀

9.7　性能的多种度量

在 9.4 节～9.6 节，我们给出的方法是，构建有单一性能度量的置信区间。然而，对大多数实际仿真来说，要同时关注的是性能的若干个度量。假定 I_s 为性能度量 u_s 的百分之 $100(1-\alpha_s)$置信区间(其中 $s=1, 2, \cdots, k$，u_s 可能全部为终止型仿真的性能度量，也可能全部为非终止型仿真的性能度量)，则所有 k 个置信区间同时包含其各自真实度量的概率满足(参见习题 9.31)

$$P(\mu_s \in I_s, s=1,2,\cdots,k) \geqslant 1-\sum_{s=1}^{k}\alpha_s \tag{9.11}$$

无论 I_s 是否独立。这个结果称为邦费罗尼(Bonferroni)不等式，对仿真研究来说具有十分重要的意义。例如，假定人们对于所有 s，对性能的 10 个不同的度量，构建 90%置信区间，即 $\alpha_s=0.1$，那么，只能声明这 10 个置信区间的每一个包含其真实度量的概率大于或等于 0。因此，人们在解释这类研究的结果时必须小心。我们刚才描述的困难在统计学文献中称为多重比较问题。

我们现在介绍当 k 值比较小时解决上述的问题的实用方法。如果人们希望 k 个置信区间的总置信水平至少为百分之 $100(1-\alpha)$，则选择各个 α_s 使得 $\sum_{s=1}^{k}\alpha_s=\alpha$(注意，$\alpha_s$ 不必相等。因此，与较重要的度量对应的 α_s 可取较小值)。因此，人们可以构建 10 个 99%置信区间，而总置信水平至少为 90%。这种方法的困难之处在于，如果采用固定样本长度法，置信区间将大于它们原来的；或者如果采用序贯法，就会需要更多的数据以达到指定的一组 k 个相对误差。基于此，我们建议 k 不大于 10。

如果性能度量数目非常大，唯一可求助的方法就是构建一般的 90%或 95%置信区间，但要意识到这些区间的一个或多个有可能并不包含其真实度量值。

例 9.34　考虑例 9.1 有 5 个服务台和一个队列的银行系统。表 9.13 给出了使用 10 次(终止型)仿真的重复运行以及式(9.1)的结果，以构建下面每个性能度量的 96.667%置信区间：

$$E\left[\frac{\int_0^T Q(t)\,dt}{T}\right],\quad E\left(\frac{\sum_{i=1}^{N}D_i}{N}\right),\quad E\left[\frac{\sum_{i=1}^{N}I_i(0,5)}{N}\right]$$

所以，总置信水平至少为 90%(参见习题 9.38)。

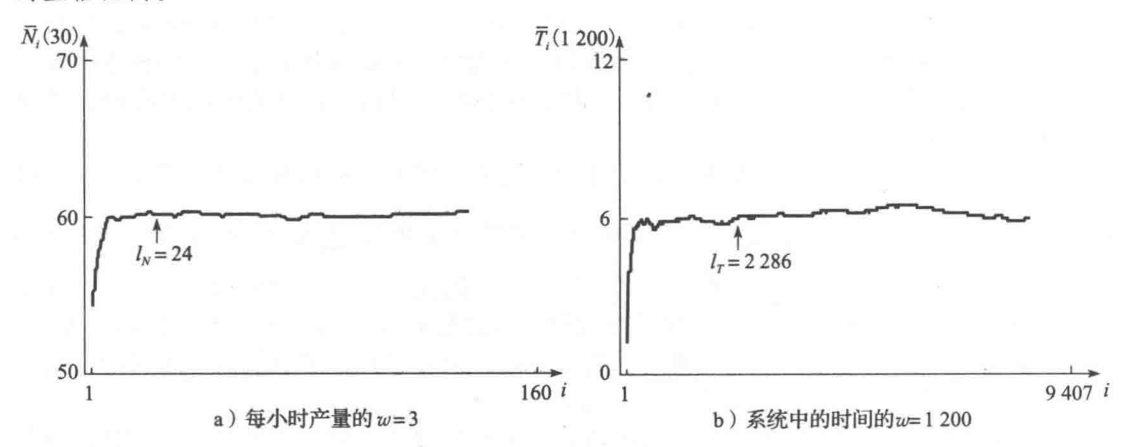
表 9.13　有 5 个服务台和一个队列的银行模型，进行 10 次重复运行的结果

性能的度量	点估计	96.667%置信区间
$E\left[\frac{\int_0^T Q(t)\,dt}{T}\right]$	1.97	[1.55，2.40]
$E\left(\frac{\sum_{i=1}^{N}D_i}{N}\right)$	2.03	[1.59，2.47]
$E\left[\frac{\sum_{i=1}^{N}I_i(0,5)}{N}\right]$	0.85	[0.80，0.90]

◀

例 9.35　假定对于例 9.25 的小型工厂，我们希望得到两个性能指标的点估计和置信区间：稳态平均每小时的产量 ν_N 与工件在系统中的稳态平均时间 ν_T，总置信水平至少为 90%。因此，我们将使每一个置信区间的置信水平为 95%。利用例 9.30 中的 10 次重复运

行，我们绘制了系统中的时间过程 T_1，T_2，…的移动平均 $\overline{T}_i(w)(i=1, 2, \cdots)$的图形，以确定其预热周期(这里的 T_i 是第 i 个分离的工件在系统中的时间)。由于该图形波动大，所以我们又进行了长度为 160 小时的 10 次重复运行，并利用整个 20 次重复运行进行分析。在图 9.21a 中，我们画出了 $w=30$ 的每小时产量的移动平均 $\overline{N}_i(w)$，而在图 9.21b 中画出了 $w=1\,200$ 的系统中的时间的移动平均 $\overline{T}_i(w)$(注意，在 160 小时的仿真运行中 T_i 观测值的个数是一个随机变量，其均值大约为 9 600。因此，对我们的分析来说，使用 20 次试验中任何一次的观测值个数的最小值，即 9 407)。根据图 9.21a 和图 9.21b，可决定预热周期分别是 $l_N=24$ 小时，$l_T=2\,286$ 次。但是，注意 2 286 次对应的是将近 38 小时。因为 24 和 2 286 分别较 160 和 9 407 小得多，我们将使用这些相同的重复运行来构建我们的置信区间。



a）每小时产量的 $w=3$　　b）系统中的时间的 $w=1\,200$

图 9.21　小型工厂的移动平均

令

$$X_j=\frac{\sum_{i=25}^{160}N_{ji}}{136},\qquad j=1,2,\cdots,20$$
$$Y_j=\frac{\sum_{i=2\,287}^{9\,407}T_{ji}}{7\,121}$$

则 ν_N 和 ν_T 的点估计和 95%置信区间分别为：

$$\hat{\nu}_N=\overline{X}(20)=60.03,\quad \hat{\nu}_T=\overline{Y}(20)=6.16\text{ 分钟}$$

以及

$$\overline{X}(20)\pm t_{19,0.975}\sqrt{\frac{0.70}{20}}=60.03\pm 0.39$$

$$\overline{Y}(20)\pm t_{19,0.975}\sqrt{\frac{0.55}{20}}=6.16\pm 0.35$$

因此，我们至少有 90%的置信度相信 ν_N 和 ν_T 同时分别落在区间[59.64, 60.42]和[5.81, 6.51]中。◄

Charnes(1995)对构建多性能度量的置信区间(或区域)的其他方法进行了综述。

9.8 重要变量的时距图

在本章，我们已经看到如何构建多种性能度量的点估计和置信区间，重点是平均系统响应。虽然这些性能度量显然相当有用，但是有些情形下，我们需要更好地表示系统性能

如何随时间变化而动态变化的。当系统的特征变化(例如可用工人的数量)是时间的函数时，这一点尤其如此。虽然动画(参见第 3.4.3 小节)对系统的短时间内的动态行为特性能够提供深刻的理解，但是这种方式并不给我们提供有关系统性能随仿真的整个长度变化的易于解释的记录。另一方面，绘出某一个或多个关键变量随仿真过程变化的图形是增强理解系统的长期动态行为特性的捷径。例如，队列长度随时间变化而变化的图形能够提供相应服务台(或多个服务台)是否具有足够的处理能力，以及等待队列所需的场地空间或能力的信息。下面的例子说明时矩图的用法，第 13 章给出了更多应用(也可参见习题 9.28)。

例 9.36 考虑例 9.31 中讨论的带午休的小型工厂。在图 9.22a 和图 9.22b 中，对于我们基于 10 次可用的仿真重复运行的第一次数据，按 30 分钟的时间增量采样，分别绘制出了机床和检测站队列中的数量。请注意，由于有半小时的午休，检测站图形有周期性的行为特性。

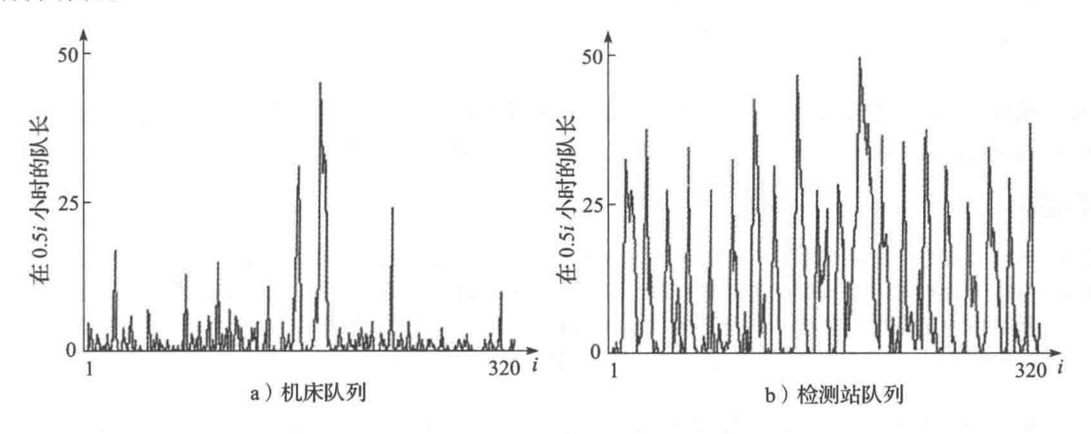


图 9.22　时间增量为 30 分钟的队列中的数量的时矩图(第 1 次运行)，小型工厂 ◀

附录 9A　期望比与对折估计

本章大部分内容是关于估计某个单一随机变量 X 的期望值，即 $E(X)$。然而，正如下面的例子所表明的那样，在仿真中有很多情况所关心的是估计两个均值的比值，如 $E(Y)/E(X)$：

(1) 对于再生法，我们在第 9.5.3 小节中看到稳态参数能够表示成两个期望值的比值。

(2) 对于例 9.5 的战争仿真，有时人们感兴趣的是估计 $E(R)/E(B)$，其中 R 和 B 分别是在战争中红军和蓝军的伤亡人数。

(3) 对于例 9.14 的银行仿真，令 $P=\sum_{i=1}^{N} D_i$ 是一天中接受服务的所有顾客的总延误时间。于是，人们感兴趣的是估计 $E(P/N)$，其含义可理解为每个顾客的平均延误时间的期望值，其中该期望值是对所有可能的天而言的。然而，也有可能人们感兴趣的是估计所有顾客的长期平均延误时间，这可以证明等于 $E(P)/E(N)$。

但是，期望比的估计值往往有偏。这里我们讨论一种获得小偏的点估计以及另一种置信区间的方法。

假定希望由数据 Y_1，Y_2，…，Y_n 和 X_1，X_2，…，X_n 估计比值 $\varphi=E(Y)/E(X)$，其中 X_i 为独立同分布的随机变量，Y_i 为独立同分布的随机变量，并且对于 $i\neq j$ 有 $\mathrm{cov}(Y_i, X_j)=0$。φ 的经典点估计为 $\hat{\varphi}_C(n)=\overline{Y}(n)/\overline{X}(n)$，关于 φ 的经典置信区间，请参见第 9.5.3 小节中的再生法的讨论。我们现在讨论 φ 的点估计和置信区间的对折法[参见 Iglehart(1975)以及 Miller(1974)]。首先定义：

$$\theta_g = n\hat{\varphi}_C(n) - (n-1)\frac{\sum_{\substack{j=1\\j\neq g}}^{n} Y_j}{\sum_{\substack{j=1\\j\neq g}}^{n} X_j}, \quad g = 1,2,\cdots,n$$

则 φ 的对折点估计为 $\hat{\varphi}_J(n) = \sum_{g=1}^{n}\theta_g/n$，一般说来，该值的偏差小于$\hat{\varphi}_C(n)$。令

$$\hat{\sigma}_J^2(n) = \frac{\sum_{g=1}^{n}[\theta_g - \hat{\varphi}_J(n)]^2}{n-1}$$

则可以证明[参见 Miller(1974)]

$$\frac{\hat{\varphi}_J(n) - \varphi}{\sqrt{\hat{\sigma}_J^2(n)/n}} \xrightarrow{\mathscr{D}} N(0,1), \quad n \to +\infty$$

该式给出了 φ 的对折百分之 $100(1-\alpha)$ 置信区间为$\hat{\varphi}_J(n) \pm z_{1-\alpha/2}\sqrt{\hat{\sigma}_J^2(n)/n}$(经典置信区间和对折置信区间的相对性能的某些实验结果请参见第 9.5.3 小节)。

习题

9.1 用启发式方法论证，使用不同随机数，由重复运行所得的可比较的输出随机变量应该是独立的。

9.2 考虑一台机器，故障停机前，其工作时间数量服从均值为 $1/\lambda$ 的指数分布。假定维修机器的时间服从均值为 $1/\omega$ 的指数分布。令 $Y(t)$表示机器在 $t(t\geqslant 0)$时刻的状态，其中，

$$Y(t) = \begin{cases} 1, & \text{如果在 } t \text{ 刻机器正在工作} \\ 0, & \text{其他} \end{cases}$$

则$\{Y(t),\ t\geqslant 0\}$为一个连续时间随机过程。进一步，可以证明[参见 Ross(2003，第 364-366 页)]：

$$P(Y(t)=1 \mid Y(0)=1) = \frac{\lambda}{\lambda+\omega}e^{-(\lambda+\omega)t} + \frac{\omega}{\lambda+\omega}$$

以及

$$P(Y(t)=1 \mid Y(0)=0) = -\frac{\omega}{\lambda+\omega}e^{-(\lambda+\omega)t} + \frac{\omega}{\lambda+\omega}$$

因此，$Y(t)$的分布取决于 t 和 $Y(0)$。在这些公式中，令 $t\to+\infty$，计算 $Y(t)$的稳态分布。该稳态分布是否还取决于 $Y(0)$?

9.3 例 9.9 中，假定条件(b)不满足。特别地，假定每一班次结束时，需要花费工人 20 分钟时间将其工具拆除；在下一班次开始，新工人需要花费 20 分钟将工具安置好。N_1，N_2，…是否具有稳态分布?

9.4 假定在例 9.9 中我们希望估计一个工件在系统中的稳态平均总时间，那么我们对制造系统进行的仿真方法是否存在问题?

9.5 确定银行系统所需服务台的个数为什么不同于确定计算机或者通信系统的硬件需求?(见例 9.10)

9.6 例 9.11 中，为何每小时产量的过程 N_1，N_2，…不存在稳态分布?

9.7 对于下列系统，你认为是终止型仿真还是非终止型仿真会更合适。在终止型情形下，说出终止事件 E。在非终止型情形下，所关心的参数是稳态参数还是稳态周期参数。

(a) 考虑一个电话系统，一个到达呼叫有可能延迟一段时间之后才获得一条线路。假定目标是估计第 100 个到达呼叫的平均延迟时间 $E(D_{100})$。

(b) 考虑一个和平时期的军事库存系统(参见第 1.5 节)，假设它长期运行。假设系统参数(如需求间隔时间分布)不随时间变化发生变化，且我们关注的是输出过程 C_1，C_2，…，其中 C_i 表示第 i 个月的总费用，进一步假定我们希望得到平均费用的度量。

(c) 考虑一个食品制造系统。发出一个生产调度单，系统生产 13 天，而然后在第 14 天时系统全部清空；然后，发出一个新的生产调度单且 2 周一重复，等等。目标是估计一个周期内的平均产量。

(d) 考虑一个提供通宵传运包裹的空运公司。装有包裹的飞机大约在晚上 11 点开始到达集散中心。包裹卸载后，按照目的地的邮政编码分类存放在仓库中。邮政编码类似的包裹被放到相同的飞机上，最后一架飞机的起飞时间大约为早上 5 点。希望估计飞机延迟起飞的平均次数。

(e) 考虑一个制造系统，其工作方式类似，每周7天。但是，假定每一天的前两个班次中，6台机床运行，但在第三个班次中，只有4台机床运行。令 N_1，N_2，…为所关注的输出过程，其中 N_i 表示第 i 个班次生产的工件数。我们关心的是平均产量的度量。你的答案是依赖于单台机床的到达速率与服务速率之间的关系吗？

9.8 对于例9.25中的小型工厂，假定系统运行5天，每天24小时，然后全部清空。因此，我们进行长度为120小时的终止型仿真，进行5次独立重复运行并构建平均周产量的点估计和95%置信区间。要想使绝对误差为50，大约需要进行多少次重复运行？要想使相对误差为5%呢？

9.9 令 p 为终止型仿真中所关注的概率，如9.4.2节所讨论的。定义独立同分布的随机变量 Y_1，Y_2，…，Y_n 满足 $\hat{p}=\overline{Y}(n)$，并将这些 Y_j 用于式(4.3)、式(4.4)和式(4.12)，以导出 p 的一个可能的置信区间。证明由式(4.4)给出的方差估计可以表示成 $\hat{p}(1-\hat{p})/(n-1)$。

9.10 考虑例9.1中的银行，利用表9.1中10次重复运行的数据构建一天中平均延误时间的分布的中位数(即0.5-分位数)的点估计，再与例9.14中的样本均值进行比较，该估计值如何？

9.11 对于例9.23的 $\rho<1$ 的 $M/M/1$ 排队模型，假定第一个顾客到达时已有的顾客数服从如下离散分布：

$$p(x)=(1-\rho)\rho^x, x=0,1,\cdots$$

它是系统中顾客数量的稳态分布。计算 D_1 的分布函数及其均值，在这种情况下，可以证明对于 $i\geqslant 2$，D_i 也具有同样的分布。

9.12 对于第9.5.1小节中的韦尔奇法，证明 $E(\overline{Y}_i)=E(Y_i)$，且 $\mathrm{var}(\overline{Y}_i)=\mathrm{var}(Y_i)/n$。

9.13 假设 Y_1，Y_2，…为协方差平稳过程，且对于 $i\geqslant 1$，有 $\rho<1$。证明，对于韦尔奇法，有 $\mathrm{var}[\overline{Y}_i(w)]<\mathrm{var}(\overline{Y}_i)$。

9.14 假定 Y_1，Y_2，…为一稳态均值等于 ν 的输出过程，$\overline{Y}(m)$ 为基于 m 个观测值的一般样本均值，请绘制 $\overline{Y}(m)$ 随 m 变化而变化的曲线，并令 l' 为超过 l' 后 $\overline{Y}(m)$ 不发生明显变化的点，则对于 $i>l'$，$E(Y_i)\approx\nu$ 以及 l' 不是非常大，在该意义下，l' 是一个好的预热周期吗？为什么？

9.15 考虑第9.5.2小节中的重复运行/删除法，证明 $E(X_i)\approx\nu$，给出两个理由说明为何式(9.5)得到的置信区间只是覆盖度意义下的近似。

9.16 考虑第9.5.2小节中的重复运行/删除法，基于使用相同的一组重复运行来确定预热周期 l 并构建置信区间，所得到的 X_j 真正是独立的吗？

9.17 对于例9.30的小型工厂，如果对于机床和检验站在 $\rho<1$ 的意义下系统得到完备的定义，则稳态平均每小时的产量应为多少？

9.18 考虑连续时间随机过程，如 $\{Q(t), t\geqslant 0\}$，其中 $Q(t)$ 是 t 时刻队列中的顾客数。假定我们希望估计队列中稳态时间平均顾客数(定义见附录1B)，使用批均值法，基于一次长度为 m 个时间单位的仿真运行。讨论两种方法，以准确获得 m 个基本离散观测值 Q_1，Q_2，…，Q_m 用于批均值法，这 m 个 Q_i 值将分批以形成 n 个批均值。

9.19 如果 Y_1，Y_2，…为协方差平稳过程，证明，对于批均值法，$C_i(k)=\mathrm{cov}[\overline{Y}_j(k), \overline{Y}_{j+i}(k)]$ 可以由下式得到：

$$C_i(k)=\sum_{l=-(k-1)}^{k-1}\frac{(1-|l|/k)C_{ik+1}}{k}$$

其中，$C_l=\mathrm{cov}(Y_i, Y_{i+l})$

9.20 令 Y_1，Y_2，…为协方差平稳过程。对于批均值法，令 $\rho_i(k)=\mathrm{cor}[\overline{Y}_j(k), \overline{Y}_{j+i}(k)]$ 且令 $b(n, k)$ 使得 $E\{\widehat{\mathrm{var}}[\overline{Y}(n, k)]\}=b(n, k)\cdot\mathrm{var}[\overline{Y}(n, k)]$。请证明，当 $k\to+\infty$ 时 $\rho_i(k)\to 0(i=1, 2, \cdots, n-1)$ 意味着当 $k\to+\infty$ 时 $E\{\widehat{\mathrm{var}}[\overline{Y}(n, k)]\}\to\mathrm{var}[\overline{Y}(n, k)]$。提示：首先证明：

$$b(n,k)=\frac{\left\{n\Big/\left[1+2\sum_{i=1}^{n-1}(1-i/n)\rho_i(k)\right]\right\}-1}{n-1}$$

9.21 对于再生法，请证明 $\nu=E(Z)/E(N)$[提示：请观察

$$\frac{\sum_{j=1}^{n'}Z_j}{\sum_{j=1}^{n'}N_j}=\frac{\sum_{i=1}^{M(n')}Y_i}{M(n')}$$

其中 n' 为再生周期的个数，且 $M(n')$ 为所有 n' 个周期中观测值(随机变量)的个数。令 $n'\to+\infty$，

并在上面的等式两端应用强大数定律]，而且有：当 $n'\to+\infty$时，$\hat{\nu}(n')=\overline{Z}(n')/\overline{N}(n')\to\nu$(以概率1)，从而，$\hat{\nu}(n')$为 ν 的强一致估计(见 9.5.3 节中 ν 的定义)。

9.22 对于例 9.32 中讨论的排队系统，在过程 D_1，D_2，…的再生点($l\geqslant 0$ 且为固定值)之后离开并离去的那些顾客的索引号正好是 l 个顾客吗？如果不是，那么他们应是在什么环境下？

9.23 对于第 1.5 节的库存的例子，请找出月费用过程的再生点序列，再次假设需求间隔时间不是指数随机变量。

9.24 假设$\hat{\nu}(n')$是基于仿真有 n'个再生周期的过程 Y_1，Y_2，…的稳态均值 ν 的(有偏)再生点估计。你是否认为可以建议用 l 个周期的预热期以减小点估计的偏差？

9.25 考虑 $\rho<1$ 的 $M/M/1$ 队列，并令一个周期内接受服务的顾客数 N 按照例 9.32 中那样来定义。按照第二个顾客的到达是第一个顾客离开前还是离开后两种情况，证明 $E(N)=1/(1-\rho)$。

9.26 对于例 9.33，请计算机床和检验站两者的利用率 ρ。应该使用怎样的到达速率？在两种情况下，该系统在 $\rho<1$ 的意义下定义是否都完备？

9.27 在例 9.33 中，如果系统定义完备，则 ν^C 应为何值？

9.28 一个制造系统有两台并行机器和单一队列组成。作业到达是速率为每小时 10 个的指数到达间隔时间，每台机器有速率为每小时 8 个的指数加工时间。每一天的前 16 个小时期间两台机器工作，但在最后 8 个小时期间只用一台。

(a) 求通过计算系统的利用率 ρ 并将其与 1 比较，请确定系统定义是否完备。

(b) 令 N_i 是第 i 个小时的产量，N_1，N_2，…是否有稳态分布？

(c) 运行 10 次重复仿真，每次长度为 480 小时(20 天)，绘制平均过程 $\overline{N}_1$，$\overline{N}_2$，…，$\overline{N}_{480}$的图形。

(d) 令 M_i 是第 i 个 24 小时天的产量。利用(c)问中得到的数据，并使用重复运行/删除法来构建稳态平均日产量 $\nu=E(M)=240$ 的点估计和 90%置信区间。

9.29 对于习题 9.28 中的系统，进行一次长度为 200 天的重复运行并令 M_i 的定义如前。利用 M_i，并使用批均值法，基于 $n=10$ 批和 $n=5$ 批两种情况来构建 $\nu=240$ 的点估计和 90%置信区间。

9.30 用标准时间序贯程序法代替批均值法重复习题 9.29。

9.31 对于 $s=1，2，\cdots，k$，令 E_s 表示以概率 $1-\alpha_s$ 发生的事件，则证明：

$$P(\bigcap_{s=1}^{k} E_s)\geqslant 1-\sum_{s=1}^{k}\alpha_s$$

其中，$\bigcap_{s=1}^{k} E_s$ 表示事件 E_1，E_2，…，E_k 的交集。不假设 E_s 是独立的[这个结果称为 Bonferroni 不等式，参见式(9.11)]。提示：通过数学归纳法证明，即首先证明$P(E_1\cap E_2)\geqslant 1-\alpha_1-\alpha_2$。然后证明，如果

$$P(\bigcap_{s=1}^{k-1} E_s)\geqslant 1-\sum_{s=1}^{k-1}\alpha_s$$

成立，则所要求的结果也成立。

9.32 对于例 9.21，为了将置信区间半长 p 减小到 0.05，请计算所需进行的重复运行的大约次数。

9.33 对于例 9.26，请证明 SP-1 和 SP-2 之间必须有两条链路才能使得该链路组的利用率不超过 0.4。

9.34 对于例 9.26，请证明 STP-A 必须有三个处理器才使得它的利用率不超过 0.4，还请证明 SP-1 必须有两个处理器才使得它的利用率小于 1。

9.35 对于例 9.26，请证明报文的总到达速率为每秒 1240 条。

9.36 对于例 9.31，STP-A 的利用率等于 0.316 是合理的吗？链路 1－2 的利用率等于 0.377 是否合理？

9.37 假定人们通过同一组重复运行构建 k 个$100(1-\alpha)$置信区间(参见 9.7 节)，则这些置信区间不独立。推导一个置信区间的期望个数的表达式，这些置信区间不包含它们各自真实的性能度量。

9.38 如果 100 个不同银行模型独立进行相应于表 9.13 所示的 10 次重复运行及后续分析，则关于大约 90 个银行的置信区间能说点什么？

第10章
比较不同的系统配置

10.1 引言

在第9章我们看到对单一系统的仿真模型的输出应用恰当的统计分析的重要性。在本章我们讨论几个不同的仿真模型的输出的统计分析，而这些模型很可能描述竞争系统的设计或不同的操作策略。这是一个非常重要的内容，因为仿真的实际作用就是在系统实现前比较这些不同。像下面的例子所阐述的那样，如果我们想要避免产生严重的错误而导致荒谬的结论并最终做出不好的决策，那么适当的统计方法是根本性的。我们希望这个例子会说明基于每个备选系统单次运行(或重复运行)的输出来做决策所固有的风险性。

例10.1 某银行计划安装一个自动取款站，必须在购买一台Zippy机器或两台Klunky机器之间做出选择。虽然购买安装和运行一台Zippy的费用是一台Klunky的2倍，但Zippy运行速度是后者的2倍。无论如何决策，由于银行的总花费是一样的，管理者希望安装的系统能够提供最好的服务。

根据现有数据，情况是在某个拥塞期，顾客每次到达1个，服从速率为每分钟1人的泊松过程。Zippy能提供的服务时间是均值为0.9分钟的独立同分布指数随机变量。换一种选择，如果安装两台Klunky，每台的服务时间是均值为1.8分钟的独立同分布指数随机变量；在此情况下，采用单一先进先出队列而不是两个分开的队列。因此，我们现在是将一个$M/M/1$队列和一个$M/M/2$队列比较，每个的利用率$\rho=0.9$，如图10.1所示。假设第一个顾客到达时系统空且闲，所关心的性能度量是前100个顾客的在队列中的平均延误期望值；对一台Zippy和两台Klunky的情形我们将(期望)量分别标记为$d_Z(100)$和$d_K(100)$(由于在队列中的等待是这个过程中最烦人的部分，且只要他们被服务就能够得到适当的安抚，银行决定忽略顾客服务时间；该问题的进一步考虑参见习题10.1)。银行大胆的系统分析人员决定对每个系统进行一次长度为100个顾客延误的仿真(采用独立随机数)并用每种情况下的100个延误的平均值来推断$d_Z(100)$和$d_K(100)$谁更小一些，并据此给出一个建议。

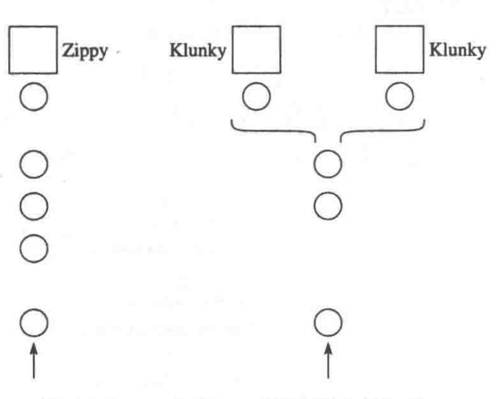


图10.1 一台Zippy还是两台Klunky?

这位分析人员给出正确的建议的可能性会有多大呢？为了解这一点，我们对分析人员的整个方案进行100次独立的实验并记录有多少次是建议了最好的系统。最好的系统实际上是安装两台Klunky，因为$d_Z(100)=4.13$，而$d_K(100)=3.70$[这些值是根据Kelton和Law(1985)的排队理论结果来确定的]。因此，我们的实验是执行两个系统的独立仿真的100对值，譬如说，将每次仿真的延误时间进行平均来获得$\hat{d}_Z(100)$和$\hat{d}_K(100)$，然后根据$\hat{d}_Z(100)$和$\hat{d}_K(100)$哪个更小来推荐Zippy或Klunky系统；部分结果如表10.1所示。在我们的100次实验中，仅48次$\hat{d}_K(100)<\hat{d}_Z(100)$，因此，分析人员做出正确决策比做出错误的决策恐怕不可能有更好的运气。

表 10.1 测试分析人员的决策规则

实验	$\hat{d}_Z(100)$	$\hat{d}_K(100)$	建议	
1	3.80	4.60	Zippy	(错)
2	3.17	8.37	Zippy	(错)
3	3.96	4.18	Zippy	(错)
4	1.91	5.77	Zippy	(错)
5	1.71	2.23	Zippy	(错)
6	6.16	4.72	Klunky	(对)
7	5.67	1.39	Klunky	(对)
⋮	⋮	⋮	⋮	
98	8.40	9.39	Zippy	(错)
99	7.70	1.54	Klunky	(对)
100	4.64	1.17	Klunky	(对)

◀

我们感觉不安的是许多仿真研究的做法类似于例 10.1 所描述的那样。困难在于仿真输出数据是随机的，因此基于每个系统的仅仅单次运行来比较两个系统是一种非常不可靠的方法。

下面的例子说明例 10.1 中的比较如何能加以改进。

例 10.2 为了阐述例 10.1 中每个一次运行方法的问题，我们在图 10.2 中的水平点图的 “$n=1$” 对中画出了所有 100 个$\hat{d}_Z(100)$和$\hat{d}_K(100)$；每个圆点(实心的或者空心的)代表在单次仿真中的 100 个延误时间的平均值，按照底部的刻度确定其位置。虽然两台 Klunky 系统的期望平均延误时间比一台 Zippy 系统的小，但观测到的平均延迟的分布有相当多的重叠。这就说明例 10.1 最后所指出的做出错误选择的概率大大增加了。

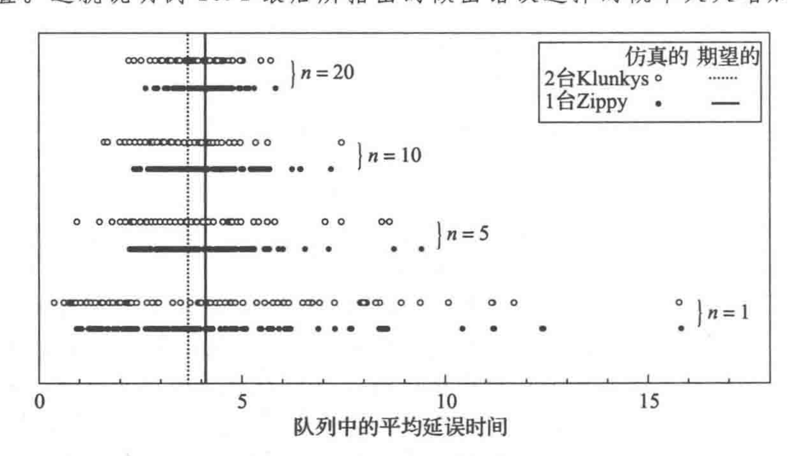


图 10.2 一台 Zippy 和两台 Klunky 对比，如例 10.1 和例 10.2 中所描述的

我们可改为对每个备选系统进行多次 n 的完全独立的重复运行，并基于它们跨重复运行的平均值来比较这些系统。特别地，对于 $j=1, 2, \cdots, n$，令 X_{1j} 为该系统的第 j 次独立重复运行在一台 Zippy 系统中的 100 个延误时间的平均值，令 X_{2j} 为该系统的第 j 次独立重复运行在两台 Klunky 系统中的 100 个延误时间的平均值(我们进行的仿真也使得 X_{1j} 和 X_{2j} 是独立的)。那么，若 $\overline{X}_1(n)$和 $\overline{X}_2(n)$分别为 X_{1j} 和 X_{2j} 的样本均值，我们可以根据较小的 $\overline{X}_i(n)$来推荐系统(因此，例 10.1 中的方法是令 $n=1$ 的特殊情形)。表 10.2，对于 $n=1, 5, 10, 20$，给出了 n 次重复运行均值的 100 个独立对中看起来一台 Zippy 系统比较好的比例，即会导致做出错误建议的比例。做出错误决策的可能性随 n 的增大而减小，

但相应的仿真成本变高。图 10.2 所示的 4 对图形也表明，随着 n 的增大，n 次重复运行均值的分布(每个圆点表示这样一个均值)密集在其期望值周围，但即使在 $n=20$，仍然存在一个相当多的重叠，其中做出错误推荐的比例仍然为 0.34。

表 10.2　例 10.2 中 n 次重复运行中错误推荐的比例

n	实验倾向于选择一台 Zippy 系统的比例
1	0.52
5	0.43
10	0.38
20	0.34

◀

例 10.1 和例 10.2 说明需要认真仔细地设计和分析比较性仿真。的确，即使每种系统设计进行 $n=20$ 次重复运行，例 10.2 表明仍然有较大的出错的可能性。第 11.2 节中将讨论一种准确比较方法，上述例子将会在该节中重新考虑，参见第 11.2.4 小节中的例 11.3。

注意，例 10.1 和例 10.2 两个涉及的都是终止型仿真(参见第 9.3 节和第 9.4 节)。正如我们在本章中将要看到的那样，用于比较备选配置的许多统计方法的一个基本需求是能搜集到独立同分布观测数据，且其期望值等于所要求的性能度量。对终止型仿真来说，这一点易于通过简单地进行独立重复运行来实现；例如，在例 10.1 和例 10.2 中的观测值的一个基本单元是在模型的一次整个重复运行中的 100 个延误时间的均值。然而，若我们希望根据稳态行为特性来比较备选的系统(参见第 9.3 节与第 9.5 节)，情况就变得更加复杂了，原因是我们不能易于得到期望值(近似)等于所要求的性能的稳态度量的独立同分布观测值。有不同的方法来处理稳态比较，对这些方法的讨论将贯穿本章，特别是在第 10.2.4 小节和第 10.4.3小节。

在本章我们的目标是给出在仿真中被发现是很有用的若干不同类型的比较和选择问题，以及适于解决这些问题的统计方法和数值例子。在本章中我们假设各种备选系统已经直接给出。在许多情况下要仔细地选择哪些系统变量要仿真；关于如何选择合适的备选系统来比较的讨论见第 12 章。

在第 10.2 节中我们探讨特殊但又很重要的情形，即只比较两个系统，通过构建它们性能度量之差的置信区间来比较。这些思想在第 10.3 节中扩展到两个以上系统的置信区间。第 10.4 节介绍一些用于选择多个备选系统中的“最好的”系统的方法，以及从备选系统中选出包含“最好的”系统的子集的方法。附录 10A 和附录 10B 讨论有关第 10.4 节的选择方法的技术问题。

10.2　两个系统的期望响应差的置信区间

这里我们考虑基于某些性能指标或期望响应来比较两个系统的情形。我们进行这种比较的办法是构建两个期望之差的置信区间，而不是进行假设检验以观察所观测的差是否显著不等于 0。鉴于假设检验仅仅得出“拒绝”或“不拒绝”的结论，而置信区间为我们除了给出这个信息(分别根据区间包含或不包含 0)外，还能够量化期望的差距有多大，如果有的话(在许多情况下，两个期望是不同的。因此，期望相等的原假设失效)。同样，我们这里将采用参数化方法，即正态理论的方法，尽管可以代之以使用非参数的类似方法[例子见文献 Conover(1999，第 281-283 页)]。参数化方法简单且常见，此外，在这种情况下应该有相当的鲁棒性，因为输出随机变量的基本分布中的令人讨厌的偏斜性(参见第 9.4 节)应该用减法加以改善(假设两个输出分布向同一个方向偏斜)。

对于 $i=1$，2，令 X_{i1}，X_{i2}，…，X_{in_i} 为系统 i 的 n_i 个独立同分布观测值的一个样本，且令 $\mu_i=E(X_{ij})$为感兴趣的期望响应；我们想要构建一个 $\zeta=\mu_1-\mu_2$ 的置信区间。X_{1j} 和 X_{2j} 是否独立依赖于仿真是如何执行的，而且可能决定了应该采用第 10.2.1 小节和

第 10.2.2 小节中讨论的两种置信区间方法中的哪一种。

10.2.1 双 t 置信区间

若 $n_1=n_2$(譬如说等于 n)，或者若对这个系统实际我们有更多的数据，我们想要丢弃一些系统的观测值，我们可以将 X_{1j} 和 X_{2j} 配对来定义 $Z_j=X_{1j}-X_{2j}(j=1, 2, \cdots, n)$。则 Z_j 是独立同分布随机变量且 $E(Z_j)=\zeta$，即我们要为之构建置信区间的量。因此，我们可令

$$\overline{Z}(n)=\frac{\sum_{j=1}^{n}Z_j}{n}$$

且

$$\widehat{\mathrm{var}}[\overline{Z}(n)]=\frac{\sum_{j=1}^{n}[Z_j-\overline{Z}(n)]^2}{n(n-1)}$$

从而构建近似 $100(1-\alpha)\%$的置信区间为：

$$\overline{Z}(n)\pm t_{n-1,1-\alpha/2}\sqrt{\widehat{\mathrm{var}}[\overline{Z}(n)]} \tag{10.1}$$

若 Z_j 是正态分布的，这个置信区间是准确的，即它覆盖 ζ 的概率是 $1-\alpha$；否则，我们依中心极限定理(参见第 4.5 节)，该定理意味着对于大的 n，该覆盖的概率接近 $1-\alpha$。这里重要的一点是我们没有假设 X_{1j} 和 X_{2j} 独立；也没有假设 $\mathrm{var}(X_{1j})=\mathrm{var}(X_{2j})$。允许 X_{1j} 和 X_{2j} 之间正相关可能是非常重要的，因为这会导致 $\mathrm{var}(Z_j)$的降低(见习题 4.13)从而有一个更小的置信区间。第 11.2 节讨论常常导致不同系统的观测值之间正相关的一种方法(公共随机数法)。式(10.1)中的置信区间将称为双-t 置信区间，且由其推导我们从本质上把双系统的问题变成了包含单一样本即 Z_j 的问题。在这个意义上，双-t 方法与第 9.4.1 小节中讨论的单一系统分析的方法相同(因此，第 9.4.1 小节中的序贯置信区间法也可以应用到这里)。重要的是，要注意 X_{ij} 是在整个重复运行上定义的随机变量；例如，X_{1j} 可以是例 10.2 中 Zippy 系统的第 j 次重复的 100 个延误时间的平均值，并不是某一个顾客的延误时间。

例 10.3 对于 1.5 节中的库存模型，假设我们想要比较在两种不同的(s, S)策略下对前 120 个月运行的每个月的期望平均总成本的影响，这里我们假设初始库存水平是 60。对于第一种策略$(s, S)=(20, 40)$，第二种策略设置$(s, S)=(20, 80)$。这里，X_{ij} 是策略 i 在第 j 次独立重复运行中每月平均总成本。我们独立地运行策略 1 和策略 2 并在每种策略下执行 $n=n_1=n_2=5$ 次独立重复运行；表 10.3 给出了仿真结果。应用双-t 方法，我们得到$\overline{Z}(5)=4.98$，$\widehat{\mathrm{var}}[\overline{Z}(5)]=2.44$，得到 $\zeta=\mu_1-\mu_2$ 的(近似)90%置信区间为[1.65, 8.31]。因此，我们有 90%置信度可以说 μ_1 与 μ_2 不同，且更进一步可以看出策略 2 更好，因为其带来了更低的平均运行成本(低在 1.65 和 8.31 之间，从假设检验不会有这样明显的结果)。我们必须用“近似”这个词来描述置信水平，因为 $n_1=n_2=5$ 对于这个模型，不一定“大”到足够使得中心极限定理有效(也可参见 10.4.1 小节中的讨论)。

表 10.3 两种库存策略的 5 次独立重复运行的每月平均总成本及其差值

j	X_{1j}	X_{2j}	Z_j
1	126.97	118.21	8.76
2	124.31	120.22	4.09
3	126.68	122.45	4.23
4	122.66	122.68	−0.02
5	127.23	119.40	7.83

◀

10.2.2　改进的双样 t 置信区间

构建 ζ 的置信区间的第二种方法是不将两个系统的观测值配对，所以 n_1 和 n_2 现在可以不同。假设 X'_{1j} 和 X'_{2j} 均为正态分布，且 X'_{1j} 独立于 X'_{2j}。在许多情况下，正态分布都是适用的，因为，X_{ij} 为许多独立观察值的平均值(例如，例 10.3 中的平均花费)。我们将给出一个由 Welch(1938)提出的旧的但是可靠的近似解决方法。

像往常一样，对于 $i=1$，2，令

$$\overline{X}_i(n_i) = \frac{\sum_{j=1}^{n} X_{ij}}{n_i}$$

且

$$S_i^2(n_i) = \frac{\sum_{j=1}^{n} [X_{ij} - \overline{X}_i(n_i)]^2}{n_i - 1}$$

则计算估计的自由度：

$$\hat{f} = \frac{[S_1^2(n_1)/n_1 + S_2^2(n_2)/n_2]^2}{[S_1^2(n_1)/n_1]^2/(n_1-1) + [S_2^2(n_2)/n_2]^2/(n_2-1)}$$

并用

$$\overline{X}_1(n_1) - \overline{X}_2(n_2) \pm t_{\hat{f},1-\alpha/2}\sqrt{\frac{S_1^2(n_1)}{n_1} + \frac{S_2^2(n_2)}{n_2}} \tag{10.2}$$

作为 ζ 的一个近似 $100(1-\alpha)\%$ 置信区间。由于 $\hat{f}$ 通常不是整数，可能必须对标准的 t 表插值，虽然统计软件通常允许非整数的自由度。式(10.2)给出的置信区间称为韦尔奇置信区间，可以用于校验现有系统的仿真模型(参见第 5.6.2 小节)。若“系统 1”是我们直接从中收集数据的实际系统，“系统 2”是我们得到仿真输出数据的相应的仿真模型，那么很可能 n_1 要远远小于 n_2。最后，如果我们比较两个被仿真的系统且希望得到一个“小的”置信区间，可以应用由 Robbins、Simons 和 Starr(1967)给出的序贯方法，该方法在最小化 n_1+n_2 最终值的情况下非常有效。它在以下意义下也渐近正确，即随着预先确定的置信区间宽度的减小，置信区间有近似正确的覆盖概率。

例 10.4 由于例 10.3 中两种不同库存策略的运行是独立来做的，我们可以应用韦尔奇法构建 ζ 的一个近似 90% 的置信区间；我们采用与表 10.3 给出的相同的 X_{ij} 数据。我们得到 $\overline{X}_1(5)=125.57$，$\overline{X}_2(5)=120.59$，$S_1^2(5)=4.00$，$S_2^2(5)=3.76$，$\hat{f}=7.99$。对标准的 t 表插值得到 $t_{7.99,0.95}=1.860$。因此，韦尔奇置信区间是[2.66，7.30]。◀

10.2.3　两种方法的对比

由于表 10.3 中的库存数据的收集满足 $n_1=n_2$ 且 X_{1j} 独立于 X_{2j}，我们既可以采用双-t 法也可采用韦尔奇法来构建 ζ 的置信区间。在这个例子中碰巧韦尔奇法的置信区间比较小，但通常我们不知道哪个置信区间会比较小。

究竟是选择双-t 法还是韦尔奇法通常因情况而定。一个要考虑的因素是公共随机数(CRN)用于仿真两个系统时(参见第 11.2 节)常常会导致 $\mathrm{var}(Z_j)$ 的过度减小，且因此得到一个非常小的置信区间；这意味着 $n_1=n_2$ 且 X_{1j} 和 X_{2j} 会不独立，所以需要应用双-t 方法。

另一方面，若 $n_1 \neq n_2$(且我们希望用到所有获得的数据)，此时应该应用韦尔奇法。这要求 X_{1j} 独立于 X_{2j}，且特别要避免使用公共随机数。注意，假设两个系统采用独立随机数的结果具有独立性，这实际上需要显式地操作才能保证，因为除非特别指出，大多数仿真软件包默认设置是相同的随机数流和种子(见第 7.2 节)。因此，按默认设置，两个系统实际上应用了相同的随机数，虽然不一定必须严格同步(见第 11.2.3 小节)，致使韦尔奇方法失效。

10.2.4 基于稳态性能度量的比较

像第10.1节中所提到的，大多数比较技术的基本要素是独立同分布观测值的一个样本，该样本具有的期望值等于做比较的性能度量。本章到目前为止的例子都是终止型仿真，所以这些观测值可以自然地通过简单重复运行仿真若干次来获得。

然而，在其他情况下，我们可能希望根据稳态性能度量(参见第9.3节和第9.5节)来比较两个(或多个)系统。这里我们不能再简单地重复运行模型，因为初始化的影响可能会使输出产生偏差，像第9.5.1小节中讨论的那样。因此，基于稳态性能度量以及第9.5节中所讨论的许多内容引起的问题进行有效的比较是更为困难的。下面两个例子说明了第9.5.2小节中描述的面向稳态分析的重复运行/删除法如何适应解决构建两个稳态均值之差的置信区间的问题。

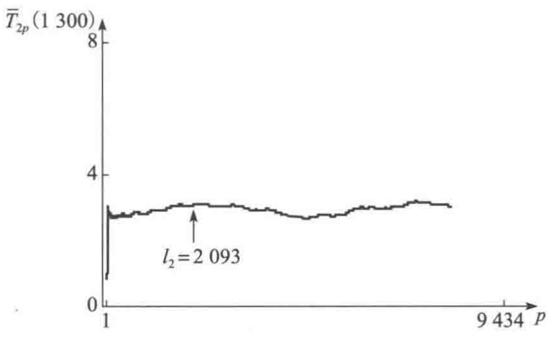


图 10.3 建议系统的系统中的时间平均值($w=1\,300$)变化

例 10.5 例9.25中的制造公司正考虑购买一套新的检测设备，希望将检测时间减少10%，因此其均匀分布在0.675分钟和0.720分钟之间。仿真研究能够有助于确定这项改变是否能够显著降低系统的稳态均值。令T_{ijp}为系统i(对应于原系统和建议的系统，i分别等于1和2)在第j次重复运行($j=1, 2, \cdots, n$)中的第p个离去的零件在系统中的时间。令l_i和m_i两者为系统i在任何重复运行中的预热期和T_{ijp}的最小个数。应用例9.33中的数据，我们得到$n=20$，$l_1=2286$，$m_1=9407$。我们随后对建议的系统($i=2$)进行20次长度为160小时的重复运行，变化的均值$\overline{T}_{2p}(1300)$如图10.3所示。从这些运行和图10.3中，我们确定了$l_2=2093$，$m_2=9434$。令

$$X_{ij}=\frac{\sum_{p=l_i+1}^{m_i} T_{ijp}}{m_i-l_i},\quad i=1,2$$

且对于$j=1, 2, \cdots, 20$，$Z_j=X_{1j}-X_{2j}$。同样，令ν_i为系统i的系统中稳态平均时间。那么，由每个系统的20的重复运行，我们应用式(10.1)以得到2.36 ± 0.31作为$\nu_1-\nu_2$的近似90%置信区间。因此，由于这个置信区间不包含0，两个稳态平均系统时间似乎是统计显著的，且反映出大约38%的降低(3.80与6.16相比，回忆一下，平均检测时间仅仅减少了10%；参见习题10.9)。 ◀

例 10.6 再次思考例9.26和例9.29中的通信网络，回忆一下，若两条链路从一个节点中出发，则选择每条链路概率为0.5。建议一种新的路由策略，报文路由到当前报文总数最小的那条链路(在传的加上队列中的)。在相同的情况下，每条链路选中的概率仍为0.5。我们进行对每项策略进行5次长度为$m=65$秒且预热期长度为$l=5$秒(见例9.26)的独立重复运行；两种策略的重复运行也彼此独立且独立于用于确定预热期的重复运行。令X_{ij}为在重复运行j对应用策略i(对应于原路由策略和建议路由策略，i分别等于1和2)在时间区间[5, 65]秒内完成的所有报文的平均端到端延迟时间，且令$Z_j=X_{1j}-X_{2j}$。同样，令ν_i为策略i的稳态平均端到端延迟时间。则我们应用式(10.1)得到[−0.12, 0.31](单位：毫秒)为$\nu_1-\nu_2$的近似90%置信区间。由于这个置信区间包含0，两个稳态均值之差不是统计显著的。我们会在第11.2节中再次讨论这个例子。 ◀

例10.5和例10.6中所用的方法本质上试图应用重复运行/删除法来获得每个系统的独立同分布观测值，其均值等于各自性能的稳态度量。第9.5.3小节中所讨论的其他几个

稳态分析的单系统方法也可考虑。例如，若预热期很长，我们可能希望对每个备选系统应用批均值法作为获取独立同分布无偏观测值的一种不同的方法。由于批均值法的成功的关键因素是消除批间的相关性，我们必须仔细地对批适当地进行定义，如在第 9.5.3 小节中讨论的那样。另一种可能的方法也许是 Chen 和 Sargent(1987)的标准化的时间序列法的两模型适配(参见第 9.5.3 小节)。

10.3　两个以上系统比较的置信区间

如果比较的只有两个系统，第 10.2 节中的方法提供了构建它们的性能度量之差的置信区间的方法。然而在许多研究中，可能有两个以上的系统，但我们仍然可以应用置信区间法。

我们会同时进行若干个置信区间声明，则它们各自的置信水平都必须调高使得所有区间覆盖其对应目标的总置信水平是在所要求的水平 $1-\alpha$ 上。我们将应用邦费罗尼(Bonferroni)不等式[参见第 9.7 节中的式(9.11)]来确保总置信水平至少为 $1-\alpha$。回想一下，邦费罗尼不等式意味着，若我们进行 c 个置信区间声明，则应该使每个各自的区间达到水平 $1-\alpha/c$，才能使得所有覆盖其目标的区间放在一起的总置信水平至少为 $1-\alpha$。例如，若我们希望构建 $c=10$ 个区间，并希望得到总置信水平为 $100(1-\alpha)\%=90\%$，我们必须使得每个单独的区间在 90%的水平上。显然，c 大，这意味着单个区间可能相当宽。

尽管为比较 k 个系统的均值有许多可以形式化的目标，但是我们在第 10.3.1 小节和第 10.3.2 小节中将主要关注两个问题：与“标准”的比较以及所有的两两比较。在第 10.3.3 小节后中我们简短描述一个不同的目标，即将每个系统与其他系统中最好的那个进行比较。其他问题和方法，请参见文献 Hochberg 和 Tamane(1987)、Hsu(1996)、Goldsman 和 Nelson(1998)以及 Swisher 等(2003)。

10.3.1　与标准比较

假设模型的变形之一是一个“标准”，可能代表现存系统或策略。若我们称该标准为系统 1，则其他变形为系统 2，3，…，k，目标是为 $k-1$ 个差 $\mu_2-\mu_1$，$\mu_3-\mu_1$，…，$\mu_k-\mu_1$ 构建 $k-1$ 个置信区间，具有总置信水平 $1-\alpha$。因此，我们构建 $c=k-1$ 个单独的区间，从而应构建它们每个水平为 $1-\alpha/(k-1)$。那么我们可以说(在至少 $1-\alpha$ 的置信水平上)，对于所有的 $i=2$，3，…，k，若区间 $\mu_i-\mu_1$ 不包括 0，则系统 i 与标准不同，若该区间包括 0，则系统 i 与标准不是显著不同的。

例 10.7　表 10.4 定义了第 1.5 节的库存系统的 $k=5$ 个不同的(s, S)策略；策略 1 和策略 2 就是用于例 10.3 和例 10.4 中的策略。假设策略 1，这里$(s, S)=(20, 40)$，是当前策略，而其他四个策略是考虑为可能的备选策略。

表 10.4　五种备选(s, S)库存策略

策略 i	s	S
1	20	40
2	20	80
3	40	60
4	40	100
5	60	100

这些备选策略哪些会与标准不同呢？为了发现这一点，我们进行每种策略的五次独立重复运行，不同策略的运行也彼此独立。单个重复运行的结果如表 10.5 所示，还有每种策略的(X_{ij})的样本均值和方差。由于要构建 $k-1=4$ 个区间，我们使每个区间的水平为 97.5%，使得总置信水平至少为 90%。表 10.6 给出了 $i=2$，3，4 与 5 的样本均值差以及

$\mu_i-\mu_1$ 的 97.5%置信区间。为了说明问题，我们应用双-t 法(第 10.2.1 小节)和韦尔奇法(第 10.2.2 小节)两种置信区间公式，由于不同模型的运行是独立的，所以两种方法都是有效的。星号表示那些区间不包含 0，也就是说，其对应的备选系统看来与标准不同。注意，这两种构建单个置信区间的方法可能会得出不同的结论；例如，$\mu_2-\mu_1$ 的双-t 区间没有表现出差异(见习题 10.10)，而韦尔奇法的置信区间表现出了差异。此外，两种方法对于小区间都没有优势。不管怎样，反正能够看出模型 4 和模型 5 与标准显著不同的(实际上，比标准更差，因为输出是运行成本)，模型 3 与标准没有不同；从结果来看，模型 2 与标准是否不同尚不清楚。

表 10.5 五种库存策略每种五次独立重复运行的每月平均总成本以及样本均值和方差

j	X_{1j}	X_{2j}	X_{3j}	X_{4j}	X_{5j}
1	126.97	118.21	120.77	131.64	141.09
2	124.31	120.22	129.32	137.07	143.86
3	126.68	122.45	120.61	129.91	144.30
4	122.66	122.68	123.65	129.97	141.72
5	127.23	119.40	127.34	131.08	142.61
均值	125.57	120.59	124.34	131.93	142.72
方差	4.00	3.76	15.23	8.79	1.87

表 10.6 对所有与标准系统进行比较的单个 97.5%置信区间($\mu_i-\mu_1$，i=2, 3, 4, 5)；* 表示显著差异

i	$\overline{X}_i-\overline{X}_1$	双 t 法		韦尔奇法	
		半长	区间	半长	区间
2	−4.98	5.45	(−10.44, 0.48)	3.54	(−8.52, −1.44) *
3	−1.23	7.58	(−8.80, 6.34)	6.21	(−7.44, 4.97)
4	6.36	6.08	(0.27, 12.46) *	4.55	(1.82, 10.91) *
5	17.15	3.67	(13.48, 20.81) *	6.15	(14.07, 20.22) *

◀

上面的例子和讨论中隐含了单个置信区间有恰当的概率[本例中是 $1-\alpha/(k-1)$]覆盖各自的目标。因此我们应该牢记第 9.4.1 小节中的鲁棒性问题。还有，由于邦费罗尼不等式是相当通用的，如何来构建单个的置信区间不重要；它们不要求结果来自每个模型相同次数的重复运行，它们也必是相互独立的。例如，我们可以尝试通过进行高方差模型的更多次重复运行来减少区间的宽度，或者通过使用公共随机数(第 11.2 节)来减少成对差的方差。此外，如第 10.2.4 小节所讨论的，通过采用为稳态差异构建单个置信区间的技术，加上由邦费罗尼不等式来调高单个置信水平，上述方法也可以应用于稳态比较。最后，人们总能解决不确定的问题(置信区间覆盖 0，正如表 10.6 中的 8 个区间，有 3 个发生的那样)，办法是进行更多的重复运行(或更长的运行时间)；然而，减少区间宽度的速度可能和进行额外的仿真一样慢——将区间宽度减半常常需要 4 倍之多的重复运行次数。

10.3.2 两两比较

在一些研究中，我们可能想要将每个系统和其他的每一个系统进行比较来检测和量化两两之间的任何显著差异。例如，不可能有一个这样的实际系统，所有 k 个备选系统都代表应该以同样的方式来对待可能的实现。一种方法会是对在 1 到 k 之间的所有 i_1 和 i_2，满足 $i_1<i_2$ 的差值 $\mu_{i_2}-\mu_{i_1}$ 构建置信区间。这里，将有 $k(k-1)/2$ 个单独的置信区间，且必须使每个区间在水平 $1-\alpha/[k(k-1)/2]$上，以使得所有区间共同的置信水平至少为$1-\alpha$。

例 10.8 现在假设表 10.4 中的五种库存策略都要应用表 10.5 中的数据进行彼此之间的比较。由于有 5(5－1)/2＝10 个可能的对，我们必须使每个单独的区间水平为 99％以达到 90％的总置信水平。表 10.7 给出了分布采用双 t 法和韦尔奇法得出的 99％区间，星号表明该区间不包含 0，即这些系统对有不同的期望运行成本。再一次注意到，这两种方法对差异是否是显著的结论并不总是一致的，没有一种方法给出的区间具有一致的更小的半长。此外，有可能得出明显矛盾的结论。例如，应用韦尔奇法我们可以得出结论是无论 μ_1 还是 μ_2 都与 μ_3 不是显著差别的，因此我们可能(自然地)想要得出"$\mu_1=\mu_3=\mu_2$"且因此从逻辑上认为"$\mu_1=\mu_2$"。但是 $\mu_1-\mu_2$ 的置信区间不包括 0，表示我们不能认为 μ_1 等于 μ_2。这里的问题是我们不敢把置信区间的声明作为相等或不相等的"证明"；在上述的讨论中我们只是不能从 μ_1 或 μ_2 与 μ_3 的比较中检测出 μ_1 与 μ_2 之间的差异，但是我们可以检测 μ_1 与 μ_2 之间的差异。这种表面上的矛盾会随着区间变小而减少，这通过进行更多次系统的重复运行，或者也许可以通过应用第 11.2 节中讨论的公共随机数法做到。

表 10.7　对所有两两比较(满足 $i_1<i_2$ 的 $\mu_{i_2}-\mu_{i_1}$)单个 99％置信区间；＊表示显著差异

		双 t 法			
		i_2			
		2	3	4	5
i_1	1	－4.98±7.18	－1.23±9.99	6.36±8.01	17.15±4.83＊
	2		3.75±9.58	11.34±8.38＊	22.12±3.80＊
	3			7.60±5.66＊	18.38±7.73＊
	4				10.78±5.85＊
		韦尔奇法			
		i_2			
		2	3	4	5
i_1	1	－4.96±4.36＊	－1.23±7.91	6.36±5.60＊	17.15±3.80＊
	2		3.75±7.86	11.34±5.88＊	22.12±3.72＊
	3			7.60±7.67	18.38±8.51＊
	4				10.78±5.89＊

◀

正如在第 10.3.1 小节的最后那样，我们要注意重要的是要确保单个置信区间有效性，在不同模型之间应用公共随机数的可能性，以及通过将适当的稳态方法用于单个置信区间来采用上述方法进行稳态比较的可能性。

10.3.3　与最好的进行多重比较

最后，我们提出另一类比较的目标，即对 k 个备选系统中的每个的均值与其他备选系统中最好的那个的均值之间的差值构建并列的置信区间，即使我们不知道其他系统中的那个实际上是最优的。这称为与最好的进行多重比较(MCB)，其目标是在 $\mu_i-\min_{l\neq i}\mu_l\ (i=1,2,\cdots,k)$ 上构建 k 个并列的置信区间，假设均值越小越好(若越大越好，则用"max"代替"min")。

Hsu(1984)给出了针对 MCB 目标的一种技术，Hochberg 和 Tamane(1987)以及 Hsu(1996)是关于多重比较方法的综合性的著作。Nelson(1993)给出了一种允许使用 CRN(公共随机数，见 11.2 节)来改进效率的 MCB 的处理方法。Damerdji 和 Nakayama(1999)、Nakayama(1997，2000)以及 Yuan 和 Nelson(1993)研究了稳态 MCB 问题。

虽然 MCB 以其自身的能力非常有用，但它与第 10.4.1 小节中讨论的排序与选择过程也是密切相关的[见文献 Nelson 和 Matejcik(1995)]。

10.4 排序与选择

本节我们要考虑的目标与只进行几个备选系统之间的比较是不同的，而且是更高的。在第 10.4.1 小节中我们介绍的方法，其目标是选择 k 个系统中的一个在某种意义上作为最好的，并控制该所选系统实际上就是最好的概率。第 10.4.2 小节考虑一个不同的目标，选择 k 个系统的一个子集 m 个，使得该所选子集也具有指定的概率包含最好的系统(这两种选择方法的有效性在附录 10A 中讨论)。在第 10.4.3 小节中我们讨论更深入的问题和方法，包括基于稳态性能度量的排序和选择问题。

10.4.1 在 k 个系统中选择最好的

像第 10.2 节和第 10.3 节中那样，令 X_{ij} 为第 i 个系统第 j 次重复运行的目标随机变量，且令 $\mu_i=E(X_{ij})$。对于本节的选择问题，以及第 10.4.2 小节中的选择问题，对第 i 个系统的不同重复运行来说，X_{ij} 被假设为独立的。除了在本节后面讨论的 Nelson 和 Matejcik(1995)方法之外，不同系统的重复运行也是独立的。例如 X_{ij} 可以是例 10.7 和例 10.8 的库存模型策略 i 的第 j 次重复运行的每月平均总成本。

令 μ_{i_l} 为 μ_i 中第 l 个最小的，则 $\mu_{i_1}\leqslant\mu_{i_2}\leqslant\cdots\leqslant\mu_{i_k}$。在本节中我们的目标是选择具有最小期望响应 μ_{i_1} 的系统(若我们想要最大均值 μ_{i_k}，可以简单地将 X_{ij} 和 μ_i 的符号取反)。令“CS”表示“正确的选择”这个事件。

观测到的 X_{ij} 的内在随机性表明，我们无法绝对确保我们将做出 CS，但是我们希望能够预先指定 CS 的概率。并且，若 μ_{i_1} 和 μ_{i_2} 实际上非常接近，我们不会在意我们是否错误地选择了系统 i_2(均值为 μ_{i_2} 的系统)，因此我们需要一种方法来避免进行大量的重复运行以区分这个不重要的差异。那么，准确的问题形式化为，我们希望若 $\mu_{i_2}-\mu_{i_1}\geqslant d^*$，有 $P(\mathrm{CS})\geqslant P^*$，其中最小 CS 概率 $P^*>1/k$ 和“无差别”量 $d^*>0$ 都是由分析人员指定的。自然会问，若 $\mu_{i_2}-\mu_{i_1}<d^*$，会发生什么(d^* 是我们有关检测的所关心的最小实际差值)。下面所给出的方法有一个很好的特征，被选中的系统的期望响应至少以概率 P^* 不大于 $\mu_{i_1}+d^*$[参见文献 Kim 和 Nelson(2006a)第 18.2.3 小节]。因此，保证我们(至少以概率 P^*)避免选择了均值比最好系统超过 d^* 的更差的系统(见图 10.4)。

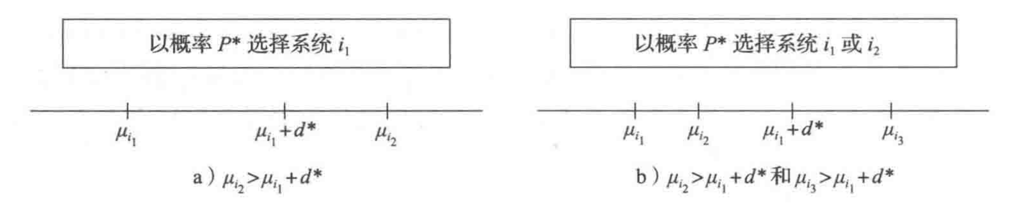


图 10.4 被选择的系统(s)

Dudewicz 和 Dalal(1975)给出了解决这个问题的统计方法，包括从 k 个系统中的每个系统“两阶段”采样。在第一个阶段，我们做每个系统的固定数量的重复运行，随后利用得到的方差估计来确定每个系统在第二个阶段需要多进行多少次重复运行以达到一个决策。必须假设 X_{ij} 是正态分布的，但(很重要的是)我们不必假设 $\sigma_i^2=\mathrm{var}(X_{ij})$ 的值是已知的；我们也不需要假设 σ_i^2 对于不同的 i 是相同的[在仿真现实系统时，假设已知方差或者方差相等(参见表 10.5)是非常不切实际的]。该方法的性能应该具有偏离正态假设的鲁棒性，特别是如果 X_{ij} 是平均值的话(当 X_{ij} 是 $M/M/1$ 排队系统中固定数量的队列中延误时间的平均值时我们已经证明了这种方法的鲁棒性)。

在第一阶段采样中，我们对 k 个系统中的每一个进行 $n_0\geqslant2$ 次重复运行，并定义第一阶段样本均值和方差分别为：

$$\overline{X}_i^{(1)}(n_0)=\frac{\sum_{j=1}^{n_0}X_{ij}}{n_0}$$

且

$$S_i^2(n_0)=\frac{\sum_{j=1}^{n_0}[X_{ij}-\overline{X}_i^{(1)}(n_0)]^2}{n_0-1}$$

其中，$i=1, 2, \cdots, k$。

随后我们计算系统 i 所需要的总样本长度 N_i 为：

$$N_i=\max\left\{n_0+1,\left\lceil\frac{h_1^2S_i^2(n_0)}{(d^*)^2}\right\rceil\right\} \tag{10.3}$$

其中，$\lceil x\rceil$是大于或等于实数 x 的最小整数；h_1(依赖于 k，P^* 和 n_0)是一个常量，可以从附录 10B 中的表 10.11 中获得。

接下来，我们对系统 $i(i=1, 2, \cdots, k)$多进行 N_i-n_0 次重复运行，并获得第二阶段样本均值为：

$$\overline{X}_i^{(2)}(N_i-n_0)=\frac{\sum_{j=n_0+1}^{N_i}X_{ij}}{N_i-n_0}$$

然后定义权重为：

$$W_{i1}=\frac{n_0}{N_i}\left[1+\sqrt{1-\frac{n_0}{N_i}\left(1-\frac{(N_i-n_0)(d^*)^2}{h_1^2S_i^2(n_0)}\right)}\right]$$

和 $W_{i2}=1-W_{i1}$

其中，$i=1, 2, \cdots, k$。

最后，定义加权样本均值为：

$$\widetilde{X}_i(N_i)=W_{i1}\overline{X}_i^{(1)}(n_0)+W_{i2}\overline{X}_i^{(2)}(N_i-n_0)$$

并选择具有最小$\widetilde{X}_i(N_i)$的系统(W_{i1}表面上很奇怪的定义的解释参见附录 10A)。

P^* 和 d^* 的选择取决于分析人员的目标和研究的特定系统；确定它们是要花费代价的，要获得一个大 P^* 或小 d^*，则 N_i 也很大，两者的计算成本互为抵消了。但是，选择 n_0 更麻烦，我们只能说，基于我们的实验以及文献中各种说法，n_0 至少为 20。若 n_0 太小，我们会得到 σ_i^2 的很差的估计 $S_i^2(n_0)$；特别是有可能 $S_i^2(n_0)$比 σ_i^2 大的多，导致 N_i 的值不必要的大；另一方面，若 n_0 太大，我们会“夸大”对某些系统必须进行重复运行的次数，这没什么意义。附录 10B 中的表 10.11 给出了在 $P^*=0.90$ 和 0.95，$n_0=20$ 和 40，$k=2, 3, \cdots, 10$ 时的 h_1 值。若对其他的 P^*、n_0 或 k 值，需要 h_1 的值，我们建议读者参考文献 Dudewicz 和 Dalal(1975)或者 Koenig 和 Law(1985)。

例 10.9 对于第 1.5 节(以及例 10.7 和例 10.8)中的库存模型，假设我们要根据前 120 个月的运行中相应的每月期望平均总成本来比较表 10.4 给出的 $k=5$ 种不同的(s, S)策略，我们用 μ_i 表示第 i 种策略的期望平均月总成本。我们的目标是选择一个具有最小 μ_i 的系统，且倘若 $\mu_{i_2}-\mu_{i_1}\geq d^*=1$，则是 $100P^*\%=90\%$确信我们做出了正确的选择。我们对每个系统执行 $n_0=20$ 次初始独立重复运行，从而由表 10.11 得到 $h=2.747$。第一阶段采样的结果在表 10.8 的 $\overline{X}_i^{(1)}(20)$和 $S_i^{(2)}(20)$列中给出。根据 $S_i^{(2)}(20)$的 h_1 和 d^*，我们随后计算每个系统的总样本长度 N_i，如表 10.8 所示。然后对每种策略再增加进行 N_i-20 次重复运行，即对策略 1 再重复运行 90 次，策略 2 再重复运行 41 次，等等，并计算第二阶段样本均值 $\overline{X}_i^{(2)}(N_i-20)$，如表 10.8 所示。最后，我们计算每个系统的权重 W_{i1} 和 W_{i2} 以及加权样本均值$\widetilde{X}_i(N_i)$。由于$\widetilde{X}_2(N_2)$是最小的加权样本均值，我们选择策略 2$(s=20,$

$S=80$)作为最低成本的配置。注意，由表 10.8 所示的 $S_i^{(2)}(20)$ 和 N_i 列，若方差估计值 $S_i^2(20)$ 高；则该方法要求一个更高的最终 N_i 值。这只是反映出这样一个事实，即系统变化越大，我们需要的数据越多。注意，若 $d^*=2$，则对策略 1 到策略 5，所需要的总重复运行次数分别为 28，21，21，21 和 21 次。

表 10.8 从 5 中库存策略中选择最好的

i	$\overline{X}_i^{(1)}(20)$	$S_i^2(20)$	N_i	$\overline{X}_i^{(2)}(N_i-20)$	W_{i1}	W_{i2}	$\widetilde{X}_i(N_i)$
1	126.48	14.52	110	124.45	0.21	0.79	124.87
2	121.92	7.96	61	121.63	0.39	0.61	121.74
3	127.16	9.45	72	126.11	0.32	0.68	126.44
4	130.71	8.25	63	132.03	0.37	0.63	131.54
5	144.07	6.20	47	144.83	0.46	0.54	144.48

◀

选择 k 个系统中最好系统的另外一种流行的方法由 Rinott(1978)提出的。这个方法使用 k 个系统通常的样本均值(基于所有第一阶段和第二阶段重复运行)来做出它的选择，而 Dudewicz 和 Dalal(D&D)方法使用 k 个系统的加权样本均值。但是，比起计算上简单一些的 Rinott 方法来，D&D 方法通常需要重复运行的次数少一些，因为 h_1 小于 Rinott 方法可与之比较的“h 值”。

* 前面关于选择最好系统所讨论的 D&D 和 Rinott 方法都假设 k 个系统是独立仿真的[⊖]。然而，在某些情况下，在仿真 k 个系统时采用 CRN 可能是有利的(根据为做出正确选择所需要的样本大小来判断)。就这一点而言，Nelson 和 Matejcik(1996)介绍了一种明确允许应用 CRN 的两阶段方法(记为 N&M)来选择最好的系统。令 $\boldsymbol{\Sigma}$ 表示随机变量 X_{1j}，X_{2j}，…，X_{kj} 的协方差矩阵(见第 6.10.1 小节)。N&M 方法假设 $\boldsymbol{\Sigma}$ 具有一种特殊的称为球形的结构，由下式定义：

$$\boldsymbol{\Sigma}=\begin{pmatrix} 2\psi_1+\tau^2 & \psi_1+\psi_2 & \cdots & \psi_1+\psi_k \\ \psi_2+\psi_1 & 2\psi_2+\tau^2 & \cdots & \psi_2+\psi_k \\ \vdots & \vdots & & \vdots \\ \psi_k+\psi_1 & \psi_k+\psi_2 & \cdots & 2\psi_k+\tau^2 \end{pmatrix}$$

其中，ψ_i 和 τ^2 是常量，且为使 $\boldsymbol{\Sigma}$ 正定，要求 $\tau^2>\sqrt{k\sum_{i=1}^{k}\psi_i^2}-\sum_{i=1}^{k}\psi_i$。球形意味着对于 $i\neq l$，有 $\mathrm{var}(X_{ij}-X_{lj})=2\tau^2$(参见习题 10.11)。这意味着系统间所有的两两差的方差是相等的，虽然边际方差和协方差可能不相等。

像 D&D 方法一样，令 n_0 为第一阶段样本长度，d^* 为无差别量，且 $p^*=1-\alpha$ 是正确选择的概率。同样，令 $g=T_{k-1,(k-1)(n_0-1),0.5}^{1-\alpha}$ 为一个自由度$(k-1)(n_0-1)$且共同相关系数为 0.5 的$(k-1)$维多变量 t 分布最大值的$(1-\alpha)$分位数。g 的值在 Bechhofer 等人(1995)的表 B.3 以及 Hochberg 和 Tamhane(1987)的表 4 中给出。那么，下面是 N&M 方法的一种表述：

(1) 在第一阶段采样中，在 k 个系统间采用 CRN，对第 i 个系统进行 $n_0\geqslant 2$ 次独立重复运行($i=1$，2，…，k)。

(2) 基于球形的假设，计算两两之差的样本方差为：

$$S^2=\frac{2\sum_{i=1}^{k}\sum_{j=1}^{n_0}(X_{ij}-\overline{X}_{i\cdot}-\overline{X}_{\cdot j}+\overline{X}_{\cdot\cdot})^2}{(k-1)(n_0-1)}$$

⊖ *第一次阅读可跳过本节的后续内容。

其中，$\overline{X}_{i\cdot}$ 表示 X_{i1}，X_{i2}，…，X_{in_0} 的样本均值；$\overline{X}_{\cdot j}$ 表示 X_{1j}，X_{2j}，…，X_{kj} 的样本均值，等等(见习题 10.12)。

(3) 计算总的所需要的样本长度 N(所有 k 个系统为常量)为：

$$N = \max\left\{n_0, \left\lceil \frac{g^2 S^2}{(d^*)^2} \right\rceil\right\} \tag{10.4}$$

(4) 在第二阶段采样中，在 k 个系统间应用 CRN，对第 i 个系统($i=1$, 2, …, k)进行 $N-n_0$ 次独立重复运行。

(5) 计算第 i 个系统的总样本均值为：

$$\overline{X}_i(N) = \frac{\sum_{j=1}^{N} X_{ij}}{N}, \quad i = 1,2,\cdots,k$$

(6) 选择具有最小 $\overline{X}_i(N)$ 的系统作为最好的备选系统。

Nelson 和 Matejcik 证明，假如 $\boldsymbol{\Sigma}$ 满足球形的性质，当 $\mu_{i_2}-\mu_{i_1}\geqslant d^*$ 时，N&M 方法正确选择的概率至少为 P^*。若 $\mu_{i_2}-\mu_{i_1}<d^*$，则返回一个系统，其均值在最好的均值的 d^* 之内。他们还证明，当协方差 σ_{ij} 为非负时，这是 CRN 假设的影响，他们的方法对偏离球状是鲁棒的。

例 10.10 考虑例 10.9 中的在五种库存策略中选择最好的这个问题，其中 $100P^*=90\%$ 且 $d^*=1$。我们对五种策略中的每一种执行 $n_0=20$ 次第一阶段的独立重复运行，应用如例 11.7 所描述的部分 CRN；特别地，我们让各种策略的需求间隔时间和需求量相同，但是独立生成交货延期。从得到的 X_{ij}，我们求得 $S^2=3.71$，并计算出所需要的总样本长度 N 为：

$$N = \max\left\{n_0, \left\lceil \frac{g^2 S^2}{(d^*)^2} \right\rceil\right\} = \max\left\{20, \frac{(1.86)^2(3.71)}{(1)^2}\right\} = 20$$

其中，$g=1.86$ 的值来自 Bechhofer 等人(1995)的表 B.3。由于 $N=20=n_0$，不必进行任何第二阶段重复运行。此外，由于五个第一阶段样本均值(即 $\overline{X}_{i\cdot}$)分别为 125.64，121.48，126.16，131.61 和 144.52，我们再次选择策略 2 为最好的策略。注意，应用 CRN 的 N&M 方法需要总共 100 次重复运行来选择策略 2 为最好的策略，而例 10.9 中，D&D 方法需要总共 353 次独立重复。因此，N&M 方法减少了将近 72%的计算量。 ◀

在第 10.3.3 小节中我们简短地讨论了与最好的进行多重比较(MCB)的问题，其目标是对于 $i=1$, 2, …, k，在 $\mu_i - \min_{l\neq i}\mu_l$ 上构建 k 个并行的置信区间。Nelson 和 Matejcik (1995)证明了大部分无差别域方法(如 D&D 和 N&M)的输出能够用于构建 MCB 置信区间，且同时既可保证正确的选择，也可保证 MCB 差值的覆盖度具有总置信水平 P^*。这种方法允许挑出具有最小均值的系统且推断出系统均值间的差值，这能够有助于基于辅助判据的决策。例如，若次优系统的均值与最好系统的均值差异不大，则由于政治或经济的原因可能会选择次优系统。

为了使这些思想更明确，再次思考 N&M 方法。那么下面的第(7)步可以附加到该方法中。

(7) 对于 $i=1$, 2, …, k，构建 $\mu_i - \min_{l\neq i}\mu_l$ 的 MCB 置信区间为：

$$\left[-(\overline{X}_{i\cdot} - \min_{l\neq i}\overline{X}_{l\cdot} - d^*)^-, (\overline{X}_{i\cdot} - \min_{l\neq i}\overline{X}_{l\cdot} + d^*)^+\right]$$

其中，$-x^-=\min(0, x)$；$x^+=\max(0, x)$。

例 10.11 对于例 10.10 中的五种库存策略，表 10.9 给出了 MCB 置信区间的计算。总的来说，我们至少有 90%置信度认为策略 2 是最好的且五个置信区间包括它们各自的 MCB 差。从第二个置信区间，我们得出结论，策略 2 不比其他策略差(上端点为 0)，它可能比其他策略便宜 \$5.16(下端点为−5.16)。其他置信区间告诉我们，规则 1，3，4 和 5 不比策略 2 好(它们区间的下端点为 0)，且可能分别贵 \$5.16，\$5.68，\$11.13 和 \$24.04。

表 10.9 五种库存策略的 MCB 置信区间

i	MCB 下端点	$\overline{X}_{i\cdot}-\min\limits_{l\neq i}\overline{X}_{l\cdot}$	MCB 上端点
1	0	4.16	5.16
2	−5.16	−4.16	0
3	0	4.68	5.68
4	0	10.13	11.13
5	0	23.04	24.04

◀

D&D 和 N&M 方法一般用于备选系统数 k 为 20 或更少的情况。这些方法的设计为，当 $\mu_{i_1}+d^*=\mu_{i_2}=\cdots=\mu_{i_k}$ 时(见附录 10A)μ_i 按最不利配置(LFC)排列，得出所期望的正确选择的概率 P^*。这是最坏情况的解决方法，它使得最好的系统尽可能难以从其他系统中区分出来，但它至少要比其他所有系统好 d^*(做出这个假设是因为这使得计算 N_i 或者 N(分别见式(10.3)和式(10.4))与真实的均值和样本的均值独立)。因此，当 k “很大”且 μ_i 相差较大时，D&D 方法和 N&M 方法可以规定比所需要的更大的样本长度以得到所要求的正确选择的概率。基于这些考虑，Nelson 等人(2001)提出了一种用于 k 很大时的筛选法。筛选阶段首先产生一个子集(大小随机)，该子集显然排除了较差系统。在随后的选择阶段对一组剩下的系统应用一个无差别域方法(如 D&D 或 N&M)来选择“最好的”系统，该组合方法保证了做出正确选择的总概率。因为对较差系统没有收集选择阶段的观测值，该组合方法需要的观测值比单独应用无差别域方法要小。这个方法的筛选部分在 Arena 仿真程序包的过程分析器中已经实现。

10.4.2 包含 k 个系统中最好系统且大小为 m 的子集的选择

现在让我们考虑一类不同的选择问题，即至少以概率 P^* 选择 k 个系统的子集正好为 m 个(m 是预先指定的)，使得所选子集将包含具有最小平均响应 μ_{i_1} 的系统。这在仿真研究初始阶段可能是一个有用的目标，在该阶段可能有大量的(k)个备选系统，且我们要执行一次初始的筛选来排除那些看起来明显是较差的系统。因此，我们可以避免花费大量的计算机时间来获得这些较差系统行为的精确估计。

我们像在第 10.4.1 小节中那样定义 X_{ij}，μ_i，μ_{i_1} 和 σ_i^2。这里，我们假设所有的 X_{ij} 是独立的和正态的(不允许 CRN)，且对于固定的 i，X_{i1}，X_{i2}，…是独立同分布的；σ_i^2 未知且不需要相等。这里，正确选择(CS)定义为选定的大小为 m 的子集包括均值为 μ_{i_1} 的系统，且我们希望，若 $\mu_{i_2}-\mu_{i_1}\geqslant d^*$，有 $P(\text{CS})\geqslant P^*$；这里，我们必须有 $1\leqslant m\leqslant k-1$，$P^*>m/k$，以及 $d^*>0$(若 $\mu_{i_2}-\mu_{i_1}<d^*$，则选定的子集包含期望响应不大于 $\mu_{i_1}+d^*$ 的系统的概率至少为 P^*)。

该方法非常类似于第 10.4.1 小节中的 D&D 方法，来自 Koenig 和 Law(1985)。我们进行第一阶段采样，对每个系统执行 $n_0\geqslant 2$ 次重复运行，并完全按照第 10.4.1 小节中那样，对于$i=1，2，\cdots，k$，定义 $\overline{X}_i^{(1)}(n_0)$和 $S_i^2(n_0)$。接下来完全按照式(10.3)计算第 i 个系统需要的总重复运行次数 N_i，只是用 h_2 替代 h_1 除外(这依赖于 m 以及 k，P^* 和 n_0)，附录 10B 中的表 10.12 可以找到[其他 m，k，P^*，n_0 可能需要的 h_2 值，请参见 Koenig 和 Law(1985)]。然后我们再执行 N_i-n_0 次重复运行，完全按照第 10.4.1 小节那样，得到第二阶段样本均值 $\overline{X}_i^{(2)}(N_i-n_0)$，权重 W_{i1} 和 W_{i2}，以及加权样本均值$\widetilde{X}_i(N_i)$。最后，我们定义所选子集由对应于 m 个最小$\widetilde{X}_i(N_i)$值的 m 个系统组成。

例 10.12 再次考虑例 10.9 中我们的五个库存系统，定义如表 10.4 所示。但是，现在假设我们希望从 $k=5$ 个系统中选择一个大小为 $m=3$ 的子集，并且若 $\mu_{i_2}-\mu_{i_1}\geqslant d^*=1$，保证选中的子集包含最好(最小成本)系统的置信水平至少为 $P^*=0.90$。再次对每个系统执行 $n_0=20$ 次初始重复运行(独立于例 10.9 中所用的那些)；该子集选择方法的完整结果

在表 10.10 中给出(由表 10.12 有 $h_2=1.243$。选中的子集由策略 1，2，3 组成)。

表 10.10 选择一个大小为 3 的包含五种库存策略中最好策略的子集

i	$\overline{X}_i^{(1)}(20)$	$S_i^2(20)$	N_i	$\overline{X}_i^{(2)}(N_i-20)$	W_{i1}	W_{i2}	$\widetilde{X}_i(N_i)$
1	124.71	17.16	27	125.64	0.80	0.20	124.89
2	121.20	12.64	21	125.69	1.01	−0.01	121.15
3	125.57	9.07	21	123.51	1.10	−0.10	125.78
4	132.39	6.22	21	133.37	1.18	−0.18	132.21
5	144.27	4.23	21	143.67	1.27	−0.27	144.43

◀

比较这里用到的 $h_2(=1.243)$和例 10.9 中用到的 $h_1(=2.747)$，我们由式(10.3)的形式看出，选择包含最好系统的大小为 3 的子集这样比较适中的目标，比选择最好的系统这样的更高的目标，需要的重复运行次数，平均起来要明显地少(事实上，第 10.4.1 小节中的选择问题只是现在的子集选择问题的 $m=1$ 的特例)。这个效果例证了我们在本节开头所提到的内容，是把这个子集选择问题当作一个相对不昂贵的初始“筛选”。

10.4.3 补充的问题和方法

10.4.1 小节讨论了从 k 个系统中选择均值最小的系统的无差别域法，其中，正确选择的概率为 $P(\mathrm{CS})\geqslant P^*$。该方法存在的一个问题是没有应用到任何第一阶段中的样本均值的信息来确定在第二阶段中要进行多少次重复试验，这使得该方法看起来效率较低。排序与选择问题我们现在讨论另外两种通用方法。Chen 等(2000)讨论了优化分配计算费(OCBA)多级(顺序)方法，鉴于对 k 个系统进行仿真的重复试验的费用总额 B，该方法将使正确选择的概率最大化。他们称分配各系统 i 的重复试验的次数应当与信噪比的平方成正比，其中，噪声是指系统 i 的样本标准方差，信号是指系统 i 的样本均值和最小样本均值之差。在该方法的第 0 阶段，每个系统重复试验 n_0 次，计算相关样本均值和方差(费用 B 的值减少了 $k\cdot n_0$ 次重复)。则在阶段 $l(l=1,2,\cdots)$，Δ 次重复被分配给 k 个系统，其中，$\Delta_i^l(\Delta_i^l\geqslant 0)$为分配到系统 i 中的重复试验且 $\sum_{i=1}^{k}\Delta_i^l=\Delta$(注意，总的分配数 Δ 在每个阶段 $l\geqslant 1$ 是相同的)。Δ_i^l 的值的选取是依据是使贝叶斯(Bayes)近似概率为 $P(\mathrm{CS})$的最大值，且优化算法应用计算所得的样本均值和方差的最新值。然后，为系统 i 指定 Δ_i^l 次附加的重复，并应用阶段 l 中所有可用的重复重新计算样本均值和样本方差。次迭代过程一直持续到剩余的重复费用达到 0 值为止(总体费用在每个阶段都减少了 Δ 次重复)。此时，总体样本均值最小的系统为最佳选择[算法详见参见 Chen 等(2010)]。

OCBA 方法和两阶段-R 法均应用了例 10.9 的库存模型，但是计划周期选定为 60 个月的而不是 120 个月来增加性能测度，以及每月平均的成本。应用 $n_0=5$，$\Delta=10$，$P^*=0.90$，$d^*=3.08(=\mu_{i_2}-\mu_{i_1})$，R 法要求 118.8 次重复运行的平均值，以及 $P(\mathrm{CS})$的估计值达到 0.967(基于 10 000 次独立试验)。另一方面，OCBA 方法在平均费用为 $B=65$ 次重复时的对 $P(\mathrm{CS})$的估计值达到了 0.969，计算效率下降了 45.3%。通常情况下，当系统的数量很大时，OCBA 方法的计算效率可以更高，这是因为在分配计算费时会有更大的灵活性。

应当注意到的是，OCBA 方法旨在使指定计算费 B 时的 $P(\mathrm{CS})$达到最大。但是，对于一个特殊的 B 值，该方法不能保证达到一特殊的 $P(\mathrm{CS})$(例如，0.90)。其他有关 OCBA 方法的文献可参见 Chen(1996)以及 Chen 和 Lee(2011)。

Chick(1997)针对排序选择问题提出了第三种通用方法，该方法是基于贝叶斯决策理论的。Chick 和 Inoue(2001a)介绍了一种新的两阶段方法，该方法旨在分配使正确选择期望概率最大化的重复，而不是无差别域法所追求的在保守的 LFC 中达到一个指定的 $P(\mathrm{CS})$。他们也同样介绍了第二个两阶段法，该方法分配重复以减少可能导致错误的期望

机会成本(两种方法均应用第一阶段的样本均值信息来确定第二阶段分配多少重复)。他们对这两种方法以及 R 方法和 NSGS 法在计划周期为 60 个月的库存模型中进行了检验，并发现新方法对 P(CS)的估计值更高。Chick 和 Inoue(2001b)提出了可以使用 CRN 的、与这两种方法相类似的方法。其他有关这些贝叶斯信息值方法(VIP0 的参考文献为 Chick (2006)；Chick 和 Frazier(2012)以及 Chick 和 Gans(2009)。

Chen 和 Kelton(2005)讨论了几种其他应用第一阶段样本均值信息决定样本长度的排序选择方法。

关于排序和选择(以及多重比较方法)的一个全面的且易于阅读的评述可以在 Swisher 等人(2003)中找到。其他通用的参考文献有 Bechhofer 等(1995)、Goldsma 和 Nelson (1998)，以及 Kim 和 Nelson(2006a)。Gray 和 Goldsman(1988)，以及 Swisher 等(2003)介绍了选择方法的应用。Boesel 等人(2003)和 Pichitlamken 等人(2005)应用排序与选择方法以提供一个统计保证，即通过递归搜算方法发现，最优系统配置(参见第 12.5 节)所返回的系统配置至少是实际仿真过的配置中最好的。在后文中讨论的有记忆的序贯选择法已经在 optquest 优化包中得到了实现(参见第 12.5.2 小节)。

子集选择

第 10.1 节和第 10.2 节中的方法应用了无差别域法，其中分析人员预先规定一个表示阈值的 d^* 值，来自不正确选择的误差低于该阈值可以认为是无关紧要的。结果是选择一个固定的、预先规定数量(可能是 1)的备选系统作为在某种意义上是“好的”系统。Koening 和 Law(1985)将其扩展到从包含 m 个最佳系统的 k 个系统中选择一个长度为 m 的子集。Chen 等(2008)讨论了解决这一问题的 OCBA 程序法。同样可参见 Chen(2009)。

相反，Gupta(1956，1965)研究出了一个方法，不需要规定无差别量(即设定 $d^*=0$)，按照预先规定的概率 P^* 产生随机大小的包含最好系统的子集。虽然对所选子集的大小未加控制，但对从大量的备选系统中筛选出那些明显不坏的系统来说，它可能是有用的第一步。Gupta 和 Santner(1973)以及 Santner(1975)扩展了这种方法，允许预先规定所选子集的最大数 m，他们还证明了这个方法与无差别域方法的关系。在仿真中这些方法的一个主要的局限性是，它们假设方差已知且相等，这在实际中不可能满足。Sullivan 和 Wilson (1989)研究了一种通用得多的限制-子集-选择法，该方法允许未知的、不相等的方差以及规定无差别量。在所有这些公式中，不坚持要求恰好选中 m 个备选项，而是定义 m 为所选子集的最大数量，其优点是，在相当清楚只有很少量的系统可能是最好的条件下，会选中的系统远远小于 m 个。最后，正如在第 10.4.1 小节中讨论的那样，Nelson 等人(2001)将一种子集选择程序与无差别域法结合起来以获得更好的统计效率。

全序贯程序法

第 10.4.1 小节和第 10.4.2 小节中讨论的排序和选择程序包括两阶段采样，其中从第一阶段观测值中计算出方差估计值并用于确定在第二阶段需要的额外观测值的数量。这种方法的一个缺点是：若第一阶段方差估计值比实际方差大很多，可能会建议第二阶段进行大量不必要的采样。因此，提出了全序贯程序法，在这里，单个“观测值”(例如，整个重复运行上的平均响应)取自在采样的当前阶段仍然在运行的每个备选系统，较差的系统在进一步的考察中被排除出去，直至识别出最好的系统为止。该程序的目标是减少按照规定的正确选择的概率选出最好系统所需要的观测值的数量。

Kim 和 Nelson(2001)开发了一种基于无差别域的序贯程序，它假设能从每个系统收集独立同分布正态观测值，且允许跨备选系统使用 CRN。Kim 和 Nelson(2007)将 R 和 KN 程序法应用于第 10.4.1 小节中规划周期为 30 个月的库存问题，其中 $n_0=10$，$P^*=0.95$，$\delta^*=1$。R 方法在两阶段内需要 1556 次重复，KN 程序法在所有阶段内共需要 232 次重复(DD 程序法针对该问题共需要 1423 次重复)。

序贯法的一个缺点是，在各替代系统所对应的仿真之间进行切换所需的时间和代价。

对此，Hong 和 Nelson(2005)研发了一个旨在平衡采样和切换成本的序贯法。同样，KN 程序法也在 Simio 仿真软件包中得到了实现(参见第 3.5.3 小节)。

之前讨论的 OCBA 程序法为顺序的(多级的)，其中，总的 Δ 次重复(观察值)在每个阶段被分配到 k 个系统中(对于在本章开始讨论过的库存示例，Δ=10 观察值在每个阶段被分配到 $k=5$ 个存货政策)。

非期望值的准则

我们之前所考虑的比较和选择方法都是基于观察一个期望的系统响应，如期望的队列中平均延误时间或期望的月平均运行成本。然而，在某些情况下，别的准则可能更合适。Goldsman(1984a，1984b)介绍了一个库存系统，其中策略 1 以 0.001 的概率获得 1 000 收益，以 0.999 的概率获得收益为 0；另一方面，策略 2 总是能获得 0.999 的收益。因此，策略 1 和策略 2 的期望收益分别是 1 和 0.999，所以据此策略 1 应该好一些。但是，策略 2 会以 0.999 的概率带来更高的收益(是 0.999 而不是 0)，所以策略 2 可能被认为是好的，即使它的期望收益较低。因此，我们可能要重新考虑我们所认为的“最好的”系统，将其定义为能够有最大概率产生“好的”结果的那个系统。

Goldsman(1984a，1984b)用这个目标审视了无差别域方法，而 Chen(1988)从子集选择的视角思考了这个问题。Miller 等人(1998)提出了一个针对这个问题的方法，该方法在达到规定概率的正确选择时需要的样本长度更小。关于这个问题的更多讨论请参见 Kim 和 Nelson (1006a)。对于选择标准的其他论述可参见 Chich 和 Inoue(2001a)以及 He 等(2007)。

备选系统间的相关性

在第 10.4.1 小节中我们看到在被仿真的系统之间使用公共随机数(CRN)能减少到达规定概率的正确选择所需要的观测值的数量。特别地，对于库存问题，使用 CRN 的 N&M 方法需要的重复运行次数比 D&D 方法少 72%，D&D 方法需要对感兴趣的系统独立地进行仿真。KN 程序法以及 Kim 和 Nelson(2001)的序贯程序法同样允许使用 CRN。另外，Fu 等(2007)和 Peng 等(2013)讨论了可以在 CRN 应用的 OCBA 程序法。

重要的是，CRN 技术事实上对 k 个系统的输出响应 X_{1j}，X_{2j}，…，X_{kj} 之间引入了所要求的正相关性，且不会“事与愿违”地引入负相关(参见第 11.2 节)。Koneig 和 Law (1982)在对库存模型进行测试选择程序中观察到事与愿违的结果，导致正确选择概率的显著下降。

一个备选系统内部的相关性

我们假设的另一种独立性是有关特定备选系统的观测值。当仿真是终止型时这并不会造成困难，因为我们只进行模型的独立重复运行，且每次重复运行产生所要求的期望响应的一个无偏估计。然而，在非终止型仿真(参见 9.3 节)的稳态参数情况下，这种无偏独立观测值并不容易得到。进行稳态参数选择的一种方法是应用重复运行/删除法产成 X_{ij}，得到的 X_{ij} 是独立的且对系统 i 的稳态均值是近似无偏的，正如在例 10.5 和例 10.6 中所做的那样。另一种可能性是进行系统 i 的单次长运行，然后令 X_{ij} 为在这次运行中第 j 批的观测值的样本均值(参见第 9.5.3 小节中批均值的讨论)；这里的关键问题是如何选择批的大小能够使得批均值近似不相关。此外，Goldsman 等人(2002)开发了两个基于每个系统一次长运行的序贯程序，其中将每个系统的基础观测值(如单个顾客的队列中延误时间)一次加一个。这些程序不允许使用 CRN。Kim 和 Nelson(2006b)给出了 KN++程序法的渐进有效性。

次级性能测度的约束条件

最近有相当多的人对受限的排序选择方法感兴趣，其目标是在 k 个系统中寻找受制于次级均值响应(性能测度)的最小(最新)均值响应。例如，安排医院的手术的目标可能是最小化平均闲置的时间资源，而其受限于病人平均等待时间的上限。解决这种问题的论文包括 Andradottir 和 Kim(2010)；Healey 等(2013)；Hunter 和 Pasupathy(2013)；Lee 等(2012)以及 Pujowidanto 等(2009)。

附录 10A 选择方法的有效性

本附录的目的是简短阐述第 10.4.1 小节和第 10.4.2 小节中的方法(N&M 方法除外)是如何证明的，以及附录 10B 中的 h_1 和 h_2 的值是如何计算的。关于更完整的讨论，我们建议感兴趣的读者参阅 Dudewicz 和 Dalal(1975)以及 Koenig 和 Law(1985)。

两个方法都是基于以下事实，对于 $i=1，2，\cdots，k$，

$$T_i = \frac{\widetilde{X}_i(N_i) - \mu_i}{d^*/h}$$

具有自由度为 n_0-1 的 t 分布，其中，h 为 h_1 或者 h_2，取决于应用哪个选择方法；T_i 也是独立的。权重 W_{il} 的表达式特意选择了一个相当特殊的形式来生成 $\widetilde{X}_i(N_i)$，使得 T_i 具有这种 t 分布[定义 W_{il} 和 $\widetilde{X}_i(N_i)$ 的另一种方式也使 T_i 具有 t 分布；参见 Dudewicz 和 Dalal (1975)]。

对于第 10.4.1 小节中的选择问题，假设 $\mu_{i_2}-\mu_{i_1}\geqslant d^*$，则当且仅当 $\widetilde{X}_{i_1}(N_{i_1})$ 是 $\widetilde{X}_i(N_i)$ 系列中最小的一个时，正确选择发生(其中 i_1 是具有最小期望响应 μ_{i_1} 的系统的索引号)。因此，若我们令 f 和 F 分别表示自由度为 n_0-1 的 t 分布的概率密度和分布函数，我们可以写出：

$$\begin{aligned}
P(\text{CS}) &= P[\widetilde{X}_{i_1}(N_{i_1}) < \widetilde{X}_{i_l}(N_{i_l}),\quad l=2,3,\cdots,k] \\
&= P\left[\frac{\widetilde{X}_{i_1}(N_{i_1})-\mu_{i_1}}{d^*/h_1} \leqslant \frac{\widetilde{X}_{i_l}(N_{i_l})-\mu_{i_l}}{d^*/h_1} + \frac{\mu_{i_l}-\mu_{i_1}}{d^*/h_1},\quad l=2,3,\cdots,k\right] \\
&= P\left(T_{i_l} \geqslant T_{i_1} - \frac{\mu_{i_l}-\mu_{i_1}}{d^*/h_1},\quad l=2,3,\cdots,k\right) \\
&= \int_{-\infty}^{+\infty} \prod_{l=2}^{k} F\left(\frac{\mu_{i_l}-\mu_{i_1}}{d^*/h_1} - t\right) f(t)\,\mathrm{d}t
\end{aligned} \tag{10.5}$$

式(10.5)的最后一行根据 $T_{i_1}=t$ 的条件和 T_i 的独立性得到。现在由于我们假设 $\mu_{i_2}-\mu_{i_1}\geqslant d^*$ 且 μ_{i_l} 随 l 的增长而递增，我们知道，对于 $l=2，3，\cdots，k$，有 $\mu_{i_2}-\mu_{i_1}\geqslant d^*$。因此，由于 F 是单调增函数，式(10.5)(改变积分中变量)得到：

$$P(\text{CS}) \geqslant \int_{-\infty}^{+\infty} [F(t+h_1)^{k-1} f(t)\,\mathrm{d}t] \tag{10.6}$$

当 $\mu_{i_1}+d^*=\mu_{i_2}=\cdots=\mu_{i_k}$ 时，式(10.6)中的等号成立，μ_i 的排列称为 LFC。因此表 10.11 是通过令式(10.6)右边的部分的积分等于 P^* 并(数值)求解 h_1 得出的。

第 10.4.2 小节中子集选择方法的有效性说明则更加复杂，但遵循类似的推理思路。我们最终能证明[参见 Koenig 与 Law(1985)]：

$$P(\text{CS}) \geqslant (k-m)\binom{k-1}{k-m}\int_{-\infty}^{+\infty} F(t+h_2)\,[F(t)]^{m-1}\,[F(-t)]^{k-m-1} f(t)\,\mathrm{d}t$$

且我们令上式的右边等于 P^* 并求解 h_2，如表 10.12 给出的。这个问题[在这里 $P(\text{CS})=P^*$]的 LFC 与第 10.4.1 小节中的问题是一样的。

附录 10B 选择方法中的常量

表 10.11 第 10.4.1 小节的方法的 h_1 值

P^*	n_0	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
0.90	20	1.896	2.342	2.583	2.737	2.870	2.969	3.051	3.121	3.182
0.90	40	1.852	2.283	2.514	2.669	2.785	2.868	2.954	3.019	3.076
0.95	20	2.453	2.872	3.101	3.258	3.472	3.472	3.551	3.619	3.679
0.95	40	2.386	2.786	3.003	3.150	3.349	3.349	3.422	3.484	3.539

表 10.12　第 10.4.2 小节的方法的 h_2 值(对于 $m=1$，应用表 10.11)

m	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$
				$P^*=0.90$，$n_0=20$				
2	1.137	1.601	1.860	2.039	2.174	2.282	2.373	2.450
3		0.782	1.243	1.507	1.690	1.830	1.943	2.038
4			0.556	1.012	1.276	1.461	1.603	1.718
				$P^*=0.90$，$n_0=20$				
5				0.392	0.843	1.105	1.291	1.434
6					0.265	0.711	0.971	1.156
7						0.162	0.603	0.861
8							0.075	0.512
9								†
				$P^*=0.90$，$n_0=40$				
2	1.114	1.570	1.825	1.999	2.131	2.237	2.324	2.399
3		0.763	1.219	1.479	1.660	1.798	1.909	2.002
4			0.541	0.991	1.251	1.434	1.575	1.688
5				0.381	0.824	1.083	1.266	1.408
6					0.257	0.693	0.950	1.133
7						0.156	0.587	0.841
8							0.072	0.497
9								†
				$P^*=0.95$，$n_0=20$				
2	1.631	2.071	2.321	2.494	2.625	2.731	2.819	2.894
3		1.256	1.697	1.952	2.131	2.267	2.378	2.470
4			1.021	1.458	1.714	1.894	2.033	2.146
5				0.852	1.284	1.539	1.720	1.860
6					0.721	1.149	1.402	1.583
7						0.615	1.038	1.290
8							0.526	0.945
9								0.449
				$P^*=0.95$，$n_0=40$				
2	1.591	2.023	2.267	2.435	2.563	2.665	2.750	2.823
3		1.222	1.656	1.907	2.082	2.217	2.325	2.415
4			0.990	1.420	1.672	1.850	1.987	2.098
5				0.824	1.248	1.499	1.678	1.816
6					0.695	1.114	1.363	1.541
7						0.591	1.004	1.252
8							0.505	0.913
9								0.430

注：† 回忆一下，对这个选择问题我们必须满足 $P^*\geq m/k$[若 $P^*=0.90$，$m=9$ 且 $k=10$，我们可以完全不需要数据选择，随机选择九个系统，就可以得到 $P(\mathrm{CS})=P^*$]。

习题

10.1　在例 10.1 和例 10.2 中，除了队列中延误时间之外，若银行确实想要计算服务时间，则应选择哪个？即假设性能度量是前 100 个顾客在系统中的期望平均总时间，而不是在队列中的期望平均延迟时间。在这种情况下哪个是“最好的”系统？哪种准则你认为更加合适？请讨论。

10.2 考虑例 10.1 中的两个系统，具有相同的初始条件和给定的性能度量，令 $\zeta = d_Z(100) - d_K(100)$。
(a) 对每个系统进行 $n_1 = n_2 = 5$ 次独立重复运行，并构建 ζ 的近似 90% 置信区间(彼此独立地进行两个系统的仿真)。应用双-t 方法。
(b) 对 Zippytel 系统进行 $n_1 = 5$ 次重复运行，对 Klunkytel 系统进行 $n_2 = 10$ 次重复运行，并构建 ζ 的近似 90% 置信区间(仍然是独立地运行两个系统)。
(c) 应用第 10.4.1 小节中的 D&D 方法选择 $k = 2$ 个系统中最好的系统，令 $n_0 = 20$，$P^* = 0.90$，$d^* = 0.4$。

10.3 对于第 2.5 节中的分时计算机模型，假设公司正考虑通过改变服务时限长度 q 来试图减少一个作业的稳态平均响应时间；考虑 q 值为 0.05，0.10，0.20 和 0.40。假设有 35 个终端，其他参数和初始条件与第 2.5 节中相同。为了获得独立同分布观测值具有近似等于一个作业的稳态平均响应时间的期望值，觉得模型预热到 50 个响应时间是恰当的，之后的 640 个响应时间取平均来获得基础 X_{ij} 观测值；那么之后需要独立地进行这 690 个响应时间的重复运行。应用第 10.4.1 小节和第 10.4.2 小节中的一种合适的选择方法(N&M 方法除外)，且 $n_0 = 20$，$P^* = 0.90$，$d^* = 0.7$ 来求解下面的每个问题。
(a) 选择四个 q 值中的最好值。
(b) 选择两个 q 值，其中一个是最好值。

10.4 对于第 2.7 节中的加工车间模型，我们现在对决定哪个工作站应该增添一台机器的问题进行一个更好的分析(参见第 2.7.3 小节，且由图 2.46 注意到，根据对现有系统单次的重复运行，工作站 1，2，4 看起来是三个最拥挤的工作站)。应用第 10.4.1 小节中的 D&D 方法给出机器添加到工作站 1，2 还是 4 的建议，假设这里只有 3 种可能的选择，令 $n_0 = 20$，$P^* = 0.90$，$d^* = 1$。应用稳态期望总平均作业总延迟时间作为性能的度量；为得到 X_{ij} 观测值，像习题 2.7 那样，预热模型 10 个八小时工作日，再应用随后的 90 个工作日的数据。将你的结论与第 2.7.3 小节最后的结论比较。根据例 10.1 的观点，整个研究可以如何进行改进?

10.5 考虑第 2.5 节中的原始分时计算机模型和习题 2.18 中描述的备选处理策略，两者都有 35 个终端。应用第 10.4.1 小节中的 D&D 方法，$n_0 = 20$，给出哪个处理策略能够得出一个作业具有最小的稳态平均响应时间的建议。这里，为得到 X_{ij}，预热模型 50 个响应时间，然后应用后面的 640 个响应时间的平均值，并进行所需要的重复运行。选择你自己的 P^* 和 d^*，也许是基于成本的考虑，或者基于你自己感觉的引起平均响应时间产生差别的“重要的”因素。

10.6 对于习题 1.22 中的制造车间，应用第 10.4.2 小节中的选择方法从 5 个 s 值(维修人员人数)中选出 3 个，其中一个得到每小时最小期望平均成本。应用 $n_0 = 20$，$P^* = 0.90$，且 $d^* = 5$。

10.7 对于习题 10.3 中的 4 个备选系统，对所有与当前($q = 0.10$)系统进行的比较构建置信区间，应用总置信水平为 90%，进行你认为需要的重复运行次数以得到的有意义的结果。

10.8 对于习题 1.22 和习题 10.6 中的制造车间，对五个 s 值的每小时期望平均成本的所有两两之差构建置信区间；应用总置信水平为 90%。按照需要执行重复运行次数以得到有意义的结论。

10.9 对于例 10.5，平均检测时间只减少了 10%，那么，如何使得在系统中的稳态平均时间能够减少 38%?

10.10 在例 10.3 中，μ_1 和 μ_2 之差是统计显著的，但是在例 10.7 中不是，为什么结果不同?

10.11 证明：球形隐含着，对于 $i \neq l$，有 $\mathrm{var}(X_{ij} - X_{lj}) = 2\tau^2$(参见第 10.4.1 小节)。

10.12 考虑第 10.4.1 小节的 N&M 方法。假设对于 $j = 1, 2, \cdots, n_0$，X_{1j}，X_{2j}，$\cdots$，X_{kj} 是多元正态分布的，协方差矩阵为 $\boldsymbol{\Sigma}$，具有球形特性，则可以证明[参见文献 Nelson 和 Matejcik(1995)中的引理 2]，S^2 得分布是 $2\tau^2 \chi^2_{(k-1)(n_0-1)} / [(k-1)(n_0-1)]$，其中，$\chi^2_\nu$ 是有 ν 自由度的 χ^2 随机变量。在这样的假设下，证明，S^2 是 $\mathrm{var}(X_{ij} - X_{lj}) = 2\tau^2$ 的一个无偏估计。

第11章
方差缩减技术

11.1 引言

贯穿本书，我们努力强调的重点之一是由随机输入驱动的仿真将产生随机输出。因此，如果要对结果进行适当地分析、解释和使用(详见第9、10、12章)，必须对仿真输出数据应用适当的统计学方法。由于大规模仿真会需要大量的计算时间和存储空间，合适的统计分析(可能需要模型多次重复运行)会变得代价相当大。有时甚至一个合适的输出的统计分析的代价可能如此之高，其结果的精度，也许是用置信区间的宽度来度量，会差到无法接受。因此，分析人员应该尝试使用任何有可能增加仿真的效率的方法。

当然，“效率”要求仔细地编程以加速执行和减少存储需求。但是，在本章中，我们关注的是统计效率，以仿真的输出随机变量的方差来度量。如果我们将感兴趣的输出随机变量的方差能减少一些(例如队列中的平均延误时间或库存系统中月平均成本)且没有影响其期望值，则我们在相同的仿真次数中可以得到更高的精度，例如更小的置信区间；或者换句话说，可以使用更少次数的仿真达到所要求的精度。某些时候，如方差缩减技术(variance reduction technique，VRT)，如果应用适当的话，能区别对待非常昂贵的仿真项目与一个简单、可用的仿真项目。

正如我们将要看到的那样，应用VRT的方法一般依赖于感兴趣的特定模型(或多个模型)。因此，为恰当地使用VRT，需要对模型的工作情况有透彻的理解。进一步，一般不可能事先知道方差减少了多大，或者(更坏的情形)与直接仿真相比，方差是否完全没减少。但是，可以进行初步运行(如果能够做的话)，以便将使用VRT的结果与直接仿真的结果进行比较；最后，一些VRT本身会增加计算成本，因此这种计算效率降低与统计效率的潜在提升必须要折中[参见Glynn和Whitt(1992a)和习题11.1]。几乎所有的VRT都需要部分分析人员的某些额外的花费(即使只是理解这项技术)，像通常的那样，这一点也必须考虑。

VRT的研究原来是在早期的计算机领域，用于蒙特卡罗仿真或者分布采样[参见第1.8.3小节，以及Hammersley和Handscomb(1964)、Margan(1984，第7章)]。然而，人们发现这些最初的VRT很多并不能直接应用于复杂动态系统的仿真。

在本章剩余部分，我们将以一定的详细程度来讨论五种通用的VRT，它们在很多仿真类型中都有很多的成功应用。对于其他VRT的详细讨论，例如分层采样和重要性采样(也可参见第11.6节)，我们推荐读者参阅Kleijnen(1974)、Morgan(1984，第7章)、Bratley，Fox和Schrage(1987，第2章)。有大量的关于VRT的文献，我们这里不力图做全面的讨论。幸运的是，有一些综述给出了很有用的VRT分类的方法，同时还包括广泛的编著人；对进一步研究该课题感兴趣的读者来说，特别推荐Wilson(1984)、Nelson(1985，1986，1987a，1987c)、L'Ecuyer(1994a)、Kleijnen(1998，第6.3.5小节和附录6.2)。此外，期刊“Management Science”(第35卷，第11期，1989年11月，由G. S. Fishman编辑)与“Association for Computing Machinery (ACM) Transactions on Modeling and Computer Simulation”(第3卷，第3期，1997年7月，由P. Glasserman和P. Heidelberger编辑)的专题对VRT的研究做出了贡献。虽然我们的讨论集中在不同VRT本身，但有可能一起使用它们，参见Kwon和Tew(1994)、Avramidis和Wilson(1996)、Yang和Liou(1996)。

11.2 公共随机数

我们考虑的第一个 VRT，公共随机数(简称 CRN)，实际上与其他 VRT 不同，它是用在我们比较两个或两个以上不同的系统配置的时候(参见第 10 章)，而不是用在研究单一系统配置的时候。虽然 CRN 方法简单，但它是所有 VRT 技术中最有用和最流行的。实际上，就像我们在第 10.2.3 小节末尾所提到的，大多数仿真程序包的默认设置都使用相同的随机数流和种子(参见第 7.2 节)，除非做了其他规定。因此，根据默认，两种配置实际上很可能使用相同的随机数，虽然未进行必要适当的同步(参见第 11.2.3 小节)，这一点对 CRN 方法的成功是很关键的。

11.2.1 基本原理

CRN 的基本思想是，我们应该在“相似的试验条件下”比较不同的配置，这样我们才能更为确信任何被观测到的性能的差异都是由于配置的差异，而非“试验条件”的变化。在仿真中，这些“试验条件”就是产生的随机变量，这些随机变量在整个仿真时间中用于驱动模型。举例来说，在排队仿真中，这些随机变量会包括到达间隔时间和顾客的服务时间；在库存仿真中，我们会包括需求间隔时间和需求量。CRN 技术得名于在很多情形都可能使用相同的基本 $U(0,1)$随机数(详见第 7 章)在整个时间驱动每一个不同的配置。但是，正如在本节的后面我们将要看到的那样，想恰当实现 CRN 往往需要某种编程技术。在经典试验设计术语中，CRN 是模块化的一种形式，即“同类比较”。CRN 还称为相关采样。

为更清楚地看看 CRN 的基本原理，考虑两个不同配置的情形，如在第 10.2 节中那样，其中，X_{1j}和 X_{2j}是第 j 次独立重复运行的第一个配置和第二个配置的观察值，且我们希望估计 $\zeta=\mu_1-\mu_2=E(X_{1j})-E(X_{2j})$。若我们对于每一系统做 n 次重复运行，并对于 $j=1,2,\cdots,n$，令 $Z_j=X_{1j}-X_{2j}$，则 $E(Z_j)=\zeta$，那么

$$\bar{Z}(n)=\frac{\sum_{j=1}^{n}Z_j}{n}$$

是 ζ 的一个无偏估计。因为 Z_j 是独立同分布的随机变量，可得

$$\operatorname{var}[\bar{Z}(n)]=\frac{\operatorname{var}(Z_j)}{n}=\frac{\operatorname{var}(X_{1j})+\operatorname{var}(X_{2j})-2\operatorname{cov}(X_{1j},X_{2j})}{n}$$

参见公式(4.5)和习题 4.13。如果两种不同配置的仿真分别独立运行，即使用不同的随机数，则 X_{1j}和 X_{2j}会是独立的，所以 $\operatorname{cov}(X_{1j},X_{2j})=0$。另一方面，如果我们能够对配置 1 和配置 2 进行某种仿真使 X_{1j} 和 X_{2j} 正相关，则 $\operatorname{cov}(X_{1j},X_{2j})>0$，这样我们的估计值 $\bar{Z}(n)$的方差将减小。因此当 $\bar{Z}(n)$是在特定仿真试验中观测到的，它的值应更接近 ζ。CRN 便是这样一种技术，这里我们通过使用(小心，正如下面第 11.2.3 小节中讨论的那样)相同的随机数来仿真两个配置，试图引入这种正相关性(这并不会改变 X_{1j}和 X_{2j}的概率分布以及，特别是，他们均值和方差)。使其成为可能的就是随机数发生器(参见第 7.1 节)的确定性、可重复生产性；不可重复产生的技术，例如通过用计算机时钟值的平方根来设定随机数发生器的种子，一般来说会排除使用 CRN 以及许多其他 VRT。

11.2.2 适用性

不幸的是，并没有一个完整的一般性的方法来证明 CRN 的有效性，即无法证明 CRN 方法它总是能减小方差。甚至即使 CRN 方法有效，我们一般也不能根据经验会预先知道我们得到的方差减小有多大的程度。CRN 方法的有效性全部完全取决于被比较的特定模型，并且它使用的前提是：分析人员(恐怕是隐含)相信不同模型对无论大的还是小的随机变量值来驱动模型都会“相似地”响应。举例来说，我们会期望在不同的排队系统的设计

中，较小的到达间隔时间都会导致每个系统有较长的延迟误时间和较长的队列。

但是，某些类的模型 CRN 的成功是有保证的。Heidelberger 和 Idlehart(1979)用某些类的再生法仿真证明了这一点，而 Bratley、Fox 和 Schrage(1987，第 2 章)对其作了进一步的研究，推导给出了 CRN 有效的条件。此类研究的更多结果可参见 Gal，Rubinstein 和 Ziv(1984)、Rubinstein，Samorodnitsky 和 Shaked(1985)，Glasserman 和 Yao(1992)。

图 11.1 所示的在原理上示意性地表示出了这一概念，此处水平轴表示用于两个仿真中的特定目的一个特定 U_k 的可能值；例如，这个 U_k 可以都用于产生服务时间。曲线表示了在其他条件都一样时，仿真结果如何作出反应。在图中的上第一行的两个仿真中的任何一个中，X_{1j} 和 X_{2j} 都以 U_k 变化相同的方向单调地做出反应，从而我们可期望 CRN 引入所希望的正相关性，并因此减小方差。但是，在底部的两个图中，X_{1j} 和 X_{2j} 以 U_k 变化相反的方向做出反应，那么 CRN 会引出入负相关性，从而出现了“事与愿违”的状况，导致 $\mathrm{cov}(X_{1j}, X_{2j})<0$，并实际上增加了方差。习题 11.2 关注考虑了这一问题的一个特殊情形。

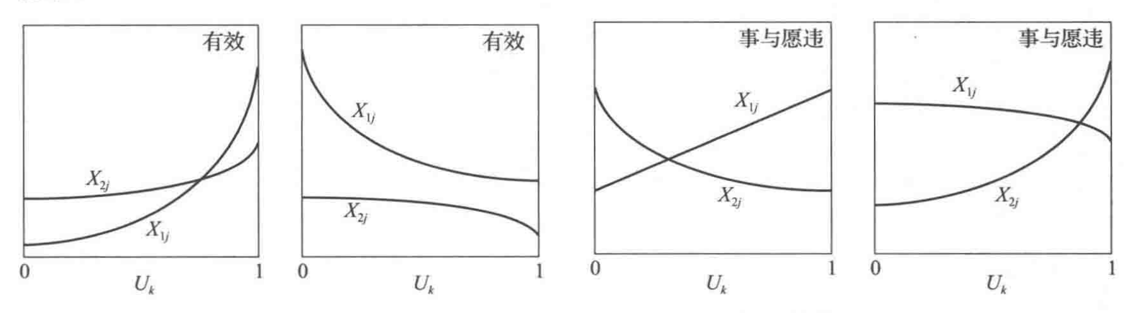


图 11.1　CRN“有效”(顶行)和“事与愿违”(底行)的模型响应

一般说来，随机数会首先用于产生服从其他分布的随机变量(参见第 8 章)，之后随机变量用于驱动仿真模型。为使 CRN 有最好的机会，我们为此首先应努力保证在这个中间变量产生步所产生的随机变量其本身对 U_k 做出单调反应，然后我们必须假设性能的度量对所产生的随机变量是单调反应的。基于这一原因，推荐使用反变换方法来产生随机变量(第 8.2.1 小节)，因为反变换方法能保证所产生的输入变量对随机数的单调性；在所有随机数变量产生方法中它更是提供了最强的正相关性[参见 Bradley(1987，第 53-54 页)或 Whitt(1976)]。然而，由于反变换方法在某些分布情形下可能计算速度较慢(可能包括使用数值方法来对分布函数求逆)，因此它计算上的低效率可能会抵消它的统计效率。基由于这一原因，Schmeiser 和 Kachitvichyanukul(1990)开发出了一种更快的非反变换方法，这种方法依旧会在所产生的随机变量中引入正相关性，正如 CRN 产生效用所需要的那样。

如果代价不高，一个试验性研究会对不同配置的 CRN 的有效性提供一个初步检查。在两个配置的情况下，使用 CRN，每一配置做 n 次重复运行，以得到输出观察值 X_{1j} 和 $X_{2j}(j=1,2,\cdots,n)$。令 $S_1^2(n)$ 和 $S_2^2(n)$ 分别是 X_{1j} 和 X_{2j} 的样本方差[使用式(4.4)]，并令 $S_Z^2(n)$ 是差值 $Z_j=X_{1j}-X_{2j}$ 的样本方差；由于该运行使用了 CRN，因此 $S_Z^2(n)$ 是 CRN 下 Z_j 的方差的无偏估计。不考虑我们使用了 CRN 这一事实，我们可知 $S_1^2(n)$ 是 $\mathrm{var}(X_{1j})$ 的无偏估计，$S_2^2(n)$ 是 $\mathrm{var}(X_{2j})$ 的无偏估计，那么，若我们进行无 CRN 的运行，$S_1^2(n)+S_2^2(n)$ 是 Z_j 的方差的无偏估计。因此，如果 CRN 是有效的，则我们可以期望观测到 $S_Z^2(n)<S_1^2(n)+S_2^2(n)$，且其差异估计了 CRN 减少了 Z_j 多少方差。当然，无论最后 CRN 是否被使用，想使用这种试验性研究结果都必须进行一些额外编程以实现 CRN。虽然有一些 CRN 事与愿违的例子，如文献 Wright 和 Ramsay(1979)、Koenig 和 Law(1982)对库存仿真所观测到的那样，但我们觉得，CRN 一般说来是一个有价值的工具，仿真分析人员面对比较两个或更多的不同配置时，应该给予认真考虑。

CRN 的另一项可能的缺点是，一些正规的统计分析方法会由于引入了相关性而可能会变得复杂。Heikes，Montgomery 和 Rardin(1976)、Kleijnen(1979)讨论了存在这种相关性时标准方差分析检验的适应性。Nelson(1987b)进一步讨论了存在 CRN 引入的相关性时有关统计分析方法的问题，他还研究了第 11.3 节中讨论的对偶变量法 VRT 产生的类似问题。对仿真试验中引入相关性，Nozari，Arnold 和 Pegden(1987)指出了在 Schruben 和 Margolin(1978)的一般框架下进行统计分析的问题，而 Tew 和 Wilson(1992)开发了检验方法，对他们的方法的适用性进行检验。Nelson 和 Hsu(1993)研究了采用第 10.3.3 小节多重比较(MCB)程序的 CRN，而 Kleijnen(1992)讨论了 CRN 对仿真的回归元模型的影响(在第 12.4 节中介绍)。有关排序与选择程序中的 CRN，读者也可参见第 10.4.3 节中的讨论，是在“选择备选配置之间的相关性”标题下。

11.2.3 同步性

要想恰当地实现 CRN，我们必须在特定的重复运行中将不同系统配置之间中的随机数进行匹配或者同步。在理想状态下，一个特定的随机数在一种配置中用于某一特定的目的，则它在所有其他配置中也用于完全相同的目的。举例来说，如果在两个不同的排队配置中，一个特定的 U_k 在两个队列配置的第一个配置中用作产生某一特定的服务时间，则它也应该用于在第二个配置中产生同一个服务时间(而不是产生到达间隔时间或者某个其他服务时间)；否则使用 CRN 的效果将丧失，甚至或者发生(最坏的情况)事与愿违的事。特别是，一般来说，仅仅直接在所有配置的仿真开始时使用相同种子的同一随机数流是不够的，它导致所有仿真都是用相同的随机数 U_1，U_2，…。下面的例子说明这一点。

例 11.1 回忆一下，在例 10.1 和例 10.2 中我们给出了自动取款机的两种竞争设计方案。第一种配置(一台 Zippy 机器)是一个 $M/M/1$ 队列；第二种配置(两台 Klunky 机器)是二个 $M/M/2$ 队列，两个队列的利用率都是 $\rho=0.9$。我们感兴趣的性能度量是最初 100 名顾客在队列中的期望平均延误时间，第一名到达的顾客发现系统到达时机器是空且闲的。因此，对于 $i=1$，2，设 X_{ij} 是 $M/M/i$ 队列第 j 次重复运行的平均延误时间。在例 10.1和例 10.2 中，我们产生了 100 个独立重复运行的 X_{1j} 和 X_{2j}，但我们要使用 CRN。

表 11.1 $M/M/1$ 和 $M/M/2$ 仿真的初始五个随机数的使用

随机数	在 $M/M/1$ 中的用途	使用时间	在 $M/M/2$ 中的用途	使用时间
U_1	A_1	0	A_1	0
U_2	A_2	A_1	A_2	A_1
U_3	S_1	A_1	S_1	A_1
U_4	A_3	A_1+A_2	A_3	A_1+A_2
U_5	A_4	$A_1+A_2+A_3$	S_2	A_1+A_2

在力图做这件事时，我们使用了单一随机数流(参见第 2.3 节和第 7.1 节)来产生到达间隔时间和服务时间两者，并在仿真第二种配置仿真开始前，我们简单地将随机数流的种子重设为其原来值。对于 $M/M/1$ 的情形，我们使用第 1.4 节中给出的编程逻辑。特别是，当一个顾客到达时，我们首先产生下一个顾客的到达时间。当顾客到达时如果服务台是空闲的，则我们立即产生顾客的服务时间并安排该顾客的离去事件。相反，如果系统服务台忙，则到达的顾客加入队列并且我们并不产生他或她的服务时间，直到该顾客在队列中的一个延误后进入服务。我们对 $M/M/2$ 队列的编程类似。为了看看随机数的使用如何失去同步的，假设，例如，顾客 2 和顾客 3 在顾客 1 离去之前到达。在表 11.1 中我们给出了最初始五个随机数在两个仿真中的使用情况，以及这些随机数的使用时间。随机数 U_1 在两个仿真中都在 0 时刻使用以产生 A_1，从而第一个到达的时间可以安排到事件表中。在两个仿真中，第一名顾客都是在 A_1 时刻到达，那么随机数 U_2 都用来产生 A_2 和安排第二

名顾客的到达时间。当第一名顾客到达时，由于两个仿真都有一个空闲的服务台，因此 U_3 用来产生 S_1 和安排第一名顾客的离去时间。第二名顾客在 A_1+A_2 时刻到达(在第一名顾客离去前)，此时 U_4 用来产生 A_3。当第二名顾客到达时，对 $M/M/2$ 仿真来说，由于还有一个空闲的服务台，因此 U_5 在此时用于产生 S_2。然而，在 $M/M/1$ 仿真中，第二名顾客到达时并没有可用的服务台，因此该顾客加入队列，且此时并不产生他或她的服务时间。在 $M/M/1$ 仿真中的下一个事件是第三名顾客在 $A_1+A_2+A_3$ 时刻到达，此时 U_5 用于产生 A_4。由上可知，U_5 在两个仿真中分别用于产生 A_4 和 S_2，随机数的使用不再同步。

在表 11.2 中我们说明了如何从 U_k 中实际产生到达间隔时间和服务时间的(回忆一下第 8.3.2 小节，一个指数随机变量是由所要求的负均值乘以随机数的自然对数来产生的)。从该表注意到，$M/M/1$ 队列的第一个服务时间与 $M/M/2$ 的第一个服务时间之间的相关性为完美的+1(参见习题 4.11)，这是最希望的。另一方面，$M/M/1$ 队列的 A_4 和 $M/M/2$ 队列的 S_2 之间的相关性也是+1。这并不是所要的效果，因为，例如，大数值的 U_5 会使 X_{1j} 更小而 X_{2j} 更大。

表 11.2　$M/M/1$ 和 $M/M/2$ 队列的 A_k 和 S_k 的计算

随机数	在 $M/M/1$ 中的用途	在 $M/M/2$ 中的用途
U_1	$A_1=-\ln U_1$	$A_1=-\ln U_1$
U_2	$A_2=-\ln U_2$	$A_2=-\ln U_2$
U_3	$S_1=-0.9\ln U_3$	$S_1=-1.8\ln U_3$
U_4	$A_3=-\ln U_4$	$A_3=-\ln U_4$
U_5	$A_4=-\ln U_5$	$S_2=-1.8\ln U_5$

◀

因此，如果我们仅仅是重复使用相同的随机数，而不关注它们是如何使用的，则我们一般并不能期望 CRN 得到恰当的实现。例 11.1 中同步性差的部分原因是因为我们特定的仿真编程做法。但是，我们并不是有意损害同步，而是代码编写的方式，看似合理实际上也正确。下面以及例 11.4 中讨论正确同步编码的问题，同时也对忽略同步的统计结果加以说明。

一般说来，保持同步的困难程度完全取决于模型的结构与参数，以及用于产生仿真中所需要的随机变量的方法。在一个给定的仿真中，可以考虑以下一些编程的“技巧”来维持同步：

- 如果有多个随机数流可用(参见第 2.3 节和第 7.1 节)，或者有多个不同的随机数发生器可以同时运行，我们可以“专用”一个流(或发生器)来为每一种特定类型的输入随机变量产生随机数。举例来说，在排队仿真中，一个流可专用来产生到达间隔时间，一个不同的流可专用来产生服务时间。流专用一般是一个好想法，且大多数仿真包提供分离的随机数流的机制(但是，常备可用的不同流的数量可能完全不足以应付大规模仿真)。此外，由于流通常只是单一随机数发生器输出的相邻段，因此都有着特定的长度，当进行长仿真或者集中重复运行时，应注意避免出现重叠现象。一个简单的计算也许粗略地告知一个流会用到多少随机数，然后就能适当地进行分配。举例来说，在单服务台排队仿真中，其中大约有 5 000 名顾客期望经通过此系统，每一名顾客都需要一个到达间隔时间和一个服务时间。如果使用反变换方法(参见第 8.2.1 小节)来产生所有这些随机变量，则我们可能需要每个流大约5 000 个随机数；如果我们重复运行此仿真 30 次，则每个流我们一共需要随机数 150 000 个左右。如果这些流长度，譬如说，为 100 000，而我们将流 1 专用于产生到达间隔时间，而流 2 专用于服务时间(像通常的做法一样)，我们看到，用于到达间隔时间的最后 50 000 个，实际上与用于服务时间的初始 50 000 个是相同的(见图 11.2)，这破坏了重复运行的独立性。

数据流 1	数据流 2	数据流 3
$U_1 \cdots\cdots U_{100\,000}$	$U_{100\,001} \cdots\cdots U_{200\,000}$	$U_{200\,001} \cdots\cdots U_{300\,000}$

间隔到达时间

服务时间

图 11.2

当然，改正的办法是跳过某些流分配：例如，保持流 1(和流 2 的前半部分)用于到达间隔时间，并使用，例如，流 6(和流 7 的前半部分)用于服务时间。

- 反变换方法产生随机变量(参见第 8.2.1 小节)可以促进同步，因为我们总是要求正好一个随机数来产生所要求的随机变量的每个值。相比之下，例如舍选法(第 8.2.4 小节)使用随机个数的 $U(0, 1)$ 随机数来产生所要求的随机变量的一个值。此外，反变换方法将随机数进行单调的变换(参见第 11.2.2 小节)，并在所产生的变量间引入最强可能的正相关性，这些变量然后成为仿真的输入，如在第 8.2.1 小节中讨论的那样；这种相关性将有望传播到输出，产生最大的方差减少。
- 在仿真某些模型中在某些特定点上“浪费”一些随机数也许是有帮助的，习题 11.4 给出了这样一个例子。
- 在某些排队仿真中，我们可以在一个顾客到达时就产生一个顾客的所有服务需求并将它们作为顾客的属性存储起来，而不是到顾客实际需要服务的时候才产生。下面的例 11.5 说明这一思想，用于第 2.7.3 小节的各种加工车间调度模型实现 CRN。另外，这种方法可以保证例 11.1 中的 $M/M/1$ 和 $M/M/2$ 的同步，防止表 11.1 中所表示出的随机数使用的混淆。然而，如果每一个实体有很多的属性，或仿真中并发实体数目变得很多时，此方法实际操作的缺点是，可能需要很大的计算机内存，而且使用此方法的困难通常与模型的特定的逻辑有关；举例来说，如果模型中在某个连接处有实体检测，可能有多种故障和多个反馈回路，也许不清楚如何预先产生这种实体所需要的所有检测时间并作为属性存储起来(参见习题 11.19)。

在使用 CRN 技术进行多次重复运行时确保重复运行的不同模型中的随机数保持同步是非常重要的，仅仅第一次是不够的。正如下面的例子将要说明的那样。

例 11.2 考虑第 1.5 节的库存模型，假设我们希望将$(s, S)=(20, 40)$，我们称为模型 1，的结果与$(s, S)=(20, 100)$，我们称为模型 2，的结果进行比较，有三个随机源：相邻需求间的时间、需求量和订单向供应商发出后的交货延期。如果我们专用数据流 1 产生需求间隔时间，数据流 2 产生需求量，数据流 3 产生交货延期，我们能保证在第一次重复运行中模型 1 和模型 2 中的所有随机数的使用恰当地同步。进一步，由于该模型的特性和逻辑的原因，我们将使用流 1 中相同个数的随机数来产生模型 1 与模型 2 两者的需求间隔时间，使用流 2 中相同个数的随机数来产生两个模型的需求量。然而，因为两个模型的再订货点 s 都是 20，但是模型 1 的订货上限 S 只有 40，而模型 2 的是 100，因此模型 1 中的订货数量要比模型 2 中的小得多，从而我们在模型 1 中发出的(小)订单数将会比模型 2 的多，在模型 1 中需要产生的交货延期数比模型 2 中的多。其结果，我们在模型 1 仿真中用掉流 3 中的随机数比我们在模型 2 仿真中的要多。所以，如果我们对两个模型开始第二次重复运行，这里流 1、流 2 和流 3 在完成模型 1 和模型 2 的第一次重复运行后都停止了，对需求时间(流 1)和需求量(流 2)来说，我们的确将使模型间取得恰当的同步，但对交货延期(流 3)来说，却不是；因此，对第二次及之后的重复运行来说，我们将不再有两个模型之间的交货延期的同步了。一个可行的补救办法是，在第一次重复运行后放弃流 1、流 2、流 3，并用(比如说)流 4、流 5、流 6 分别作为需求间隔时间、需求量和交货延迟来启动第二次重复运行，然后，第三次重复运行换到流 7、流 8、流 9，等等。假设我们在任何一次重复运行中用掉的不会多于任意一随机源的整个流，并且我们有足够的随机数流可用

于执行要求次数的重复运行，这将使得对所有的重复运行来说交货延迟回到恰当的同步，就可能加强 CRN 的效果。另一种需要较少流的补救办法是什么(参见习题 11.17)？例 11.7 讨论了交货延迟实际上是否应在两个模型间保持相同的问题。◀

应该提及的是，Arena[Kelton(2004，第 512 页)]，AutoMod[Banks(2004，第 367 页)]和 WITNESS[Lanner(2006)]仿真程序包中都有在第一次重复之后维持同步的专门特点(详见第 7.3.2 小节)。

即使人们借助了上文所讨论的各种编程技巧，但是在所研究的所有配置间达到完全同步简直是不可能的。另外，完全同步所需的额外编程开销、计算时间和存储需求，与获得的方差减小相比可能是不值得的。因此，我们或许考虑在各种配置间实现某些输入随机变量的同步，并独立地产生其他变量。举例来说，在一个复杂的排队网络中可以方便地实现到达间隔时间同步，而服务时间则不是。总之，使用 CRN 的收益以及我们实现同步的程度取决于具体情况。

11.2.4 实例

由于 CRN 的适用性、能力，以及合适的同步方法都相当依赖于模型，因此我们将给出在特定情形下的一些 CRN 应用例子。

例 11.3 我们现在重新进行例 11.1 中 $M/M/1$ 与 $M/M/2$ 的比较，但是此次我们正确地使用同步并给出实际的仿真结果。使用各自的流来实现 CRN，在仿真的 4 个 100 对数据的序列中我们估计每一个序列的各种同步级别的效果。在第一个序列中，表 11.3 中标注“I”的一列，所有运行是独立的，即完全不使用 CRN。在第二个序列中(表 11.3 中的“A”列)，两个模型的到达间隔时间是用 CRN 产生的，但我们独立产生服务时间。对于第三个序列(“S”)，到达间隔时间是独立的，但使用 CRN 产生服务时间，则两个系统将经历“相同”顺序的到达服务需求的序列(实际上 $M/M/2$ 的到达服务需求是 $M/M/1$ 的 2 倍)。最后，第四个序列(“A&S”)是完全同步的，即到达间隔时间和服务时间都是匹配的。我们可以将这四种方案在物理上理解为：

I　不同的顾客(根据他们的服务需求)到达这两个配置系统，且在不同的时间；
A　不同的顾客到达这两个配置系统，但在相同的时间；
S　相同的顾客到达这两个配置系统，但在不同的时间；
A&S　相同的顾客在相同的时间到达这两个配置系统。

表 11.3 使用 CRN 技术后 $M/M/1$ 和 $M/M/2$ 序列的统计结果

	I	A	S	A&S
$S^2(100)$	18.00	9.02	8.80	0.07
90%置信区间半长	0.70	0.49	0.49	0.04
$\hat{p}$	0.52	0.37	0.40	0.03
$\widehat{\text{cor}}(X_{1j}, X_{2j})$	−0.17	0.33	0.44	0.995

◀

由这 4 种情况的每一种的 100 对仿真数据，我们估计 $\text{var}(Z_j)$ 使用通常的无偏方差估计 $S^2(100)$ 来，将式(4.4)应用于 Z_j 可得。由此，ζ 的名义 90% 置信区间的半长是 $1.645\sqrt{S^2(100)/100}$。我们也计算了这 100 对可能做出“错误”决策，即 $X_{1j}<X_{2j}$ 的比例 $\hat{p}$[因为 $E(X_{1j})>E(X_{2j})$，正如在例 10.1 中讨论过的那样]。为更直接验证 CRN 是否引入了所要求的正相关性，我们也使用下式估计 X_{1j} 和 X_{2j} 间的相关性：

$$\widehat{\text{cor}}(X_{1j}, X_{2j}) = \frac{\frac{1}{99}\sum_{j=1}^{100}[X_{1j}-\overline{X}_1(100)][X_{2j}-\overline{X}_2(100)]}{\sqrt{S_1^2(100)S_2^2(100)}}$$

其中，$\overline{X}_i(100)$是对于所有 j 的 X_{ij} 的样本均值，$S_i^2(100)$是对于所有 j 的 X_{ij} 的样本方差(参见习题 4.29)。

由表 11.3 我们可以看到，与独立采样(I)相比，完全同步(A&S)得到的估计方差减小，这里相当明显，减小了 99%还多，或许是因为两个系统相当相似的原因。相应地，其置信区间的半长也从 0.7(I)下降到了 0.04(A&S)，降低了将近 95%。从另一个角度看，我们要问，为了使我们对 ζ 的估计 $\overline{Z}(n)$的精度(可能用置信区间的半长来度量)达到等于完全同步的 CRN 所得到的精度，在独立采样下，我们需要每个系统究竟进行多少次重复运行。如果我们对每一系统在独立采样下进行 n_I 次重复运行，则半长会近似正比于 $\sqrt{18.00/n_I}$(忽略自由度)；另一方面，如果我们在完全同步的情况下对每一系统进行 n_C 次重复运行，则半长会近似正比于 $\sqrt{0.07/n_C}$。让两个平方根相等，我们可得 $n_I/n_C=257.14$；即在独立采样下，为得到与我们用完全同步得到的精度相同，我们需要进行的重复运行次数要多 250 倍以上。

做出错误决策的估计概率从 52%降低到了 3%(参见例 10.1)。看看估计的相关性，我们看到，完全同步的 CRN 在两种配置的输出度量之间引入了非常强的相关性，这也就解释了为什么方差会有巨大的减小。对于两个部分同步的方案，我们只能得到较弱(但仍然是正)的相关性，以及相应地，方差减小较弱和 $\hat{p}$ 只有有限的下降。

本例中 CRN(完全同步)的效果可以以几种方式图形地表示出来。图 11.3a 表示在独立采样下在 100 对运行得到的各自重复运行结果对重复运行次数的图形。其中，实心的圆圈是 $M/M/1$ 模型在队列中的平均延误时间(X_{1j})，空心的圆圈是 $M/M/2$ 模型在队列中的平均延误时间(X_{2j})；对于每个固定的 j，X_{1j} 和 X_{2j} 使用垂线连接，因此其长度是 $|Z_j|$。特别是注意到，其中有 52 对，空心圆圈($M/M/2$)在线的顶部，而实心圆圈($M/M/1$)在底部，这是错误的顺序(根据它们的期望值)，与表 11.3 中的“I”列的 $\hat{p}=0.52$ 对应。图 11.3b 所示的与图 11.3a 所示的类似，但在完全同步 CRN 下，Z_j 呈现了较好的行为特性，没有图 11.3a 中出现的那种很长的垂线，且线段长度内部更加一致。这是因为有正相关时 $|X_{1j}-X_{2j}|$ 的方差更小，由于大的 X_{1j} 倾向于与一个大的 X_{2j} 相伴，而小的 X_{1j} 和 X_{2j} 值也倾向于一起出现。此外，在图 11.3b 中只有 3 个情形，垂线的顶部有 $M/M/2$ 空心圆圈，而在底部有 $M/M/1$ 的实心圆圈，相应于表 11.3 中的“A&S”列 $\hat{p}=0.03$。

图 11.4 给出一种直接观察 CRN 在本例中引入相关性的方法，其中我们图示了独立采样(空心三角形)和完全同步 CRN(实心三角形)两者的数据对(X_{1j}，X_{2j})。虽然独立对中没有明显的模式，但我们注意到 CRN 对的一条明显的直线(正斜率)，它对应于在表 11.3 中“A&S”列中的很强的正相关估计(0.995)。

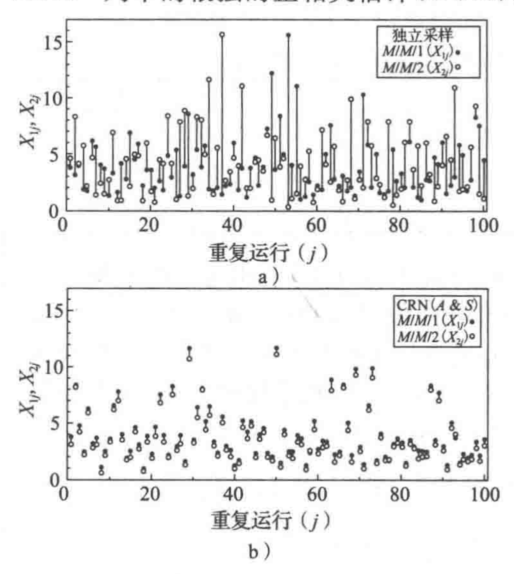


图 11.3 $M/M/1$ 与 $M/M/2$ 模型：单个重复运行

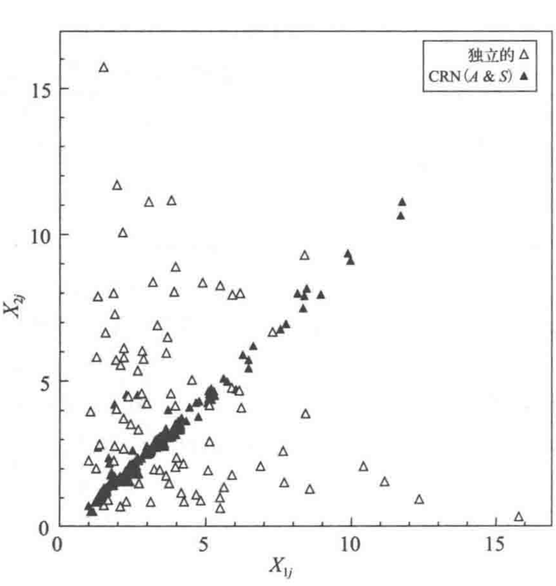


图 11.4 $M/M/2$ 平均延误时间(纵轴)与 $M/M/1$ 平均延误时间(横轴)相关性

在图 11.5 底端的“$n=1$”的点对图中，每一个方块代表一个 Z_j，按照底部所示的标尺绘出，其中空心方块是独立采样的结果，实心方块(图中它们跑到一起了)是完全同步 CRN 的结果。可以看到，CRN 的 Z_j 围绕其期望值 ζ 的分散度较独立采样情形窄得多。我们在图 11.5 中还给出了在相同标尺下当 $n=5$，10 和 20 时有关 $\overline{Z}(n)$ 的 100 次观察结果图(在独立抽样情形它们与图 10.3 中的类似情形对应)。引人注目的是，我们用 $n=1$ 的 CRN 来做的分散度比我们用独立采样高达 $n=20$ 来做还好得多。因此，CRN 在这里给我们的结果比我们花 20 倍时间使用独立采样得到的结果还要好!

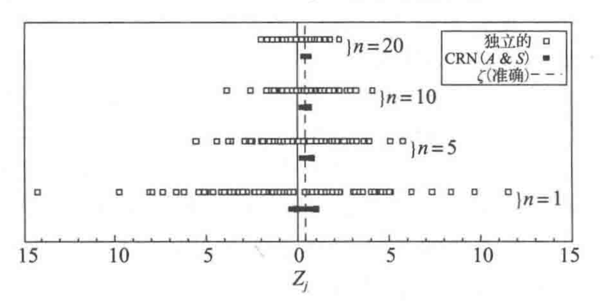


图 11.5 $M/M/1$ 与 $M/M/2$ 对比：差异

与例 11.1 不同，我们在例 11.3 中使用了正确的同步技术来实现 CRN，即对两个不同的随机源使用了各自的流。下个例子说明保持恰当同步的统计重要性。

例 11.4 我们重新运行例 11.3 的实验，假设使用完全的“A&S”CRN，但此次我们一般忽略了同步。我们的代码仍然都是“正确”的，即忠实地对两个模型进行仿真，且代表了人们实际上能够编写的程序，并没有刻意地考虑破坏同步性。在“代码 1”中我们按到达来产生服务需求，并从始至终使用同一个随机数流。事实上，这仍然获得同步，如在表 11.4 中证实的那样(表 11.4 中的数据不同于表 11.3 中的“A&S”列，因为那里使用了各自的流)。表 11.4 中的“代码 2”使用单一流，但不按到达来产生服务时间，而是等到顾客进入服务时才产生服务时间；在这种情况下，正如例 11.1 中描述的，随机数会混淆使用。正如在表 11.4 中看到的那样，结果并不比表 11.3 中“I”情形的结果好多少，这是由于我们失去了 CRN 的好处(显然接近全部失去)。“代码 3”与“代码 2”相同，只是“代码 3”中在到达子程序(详见第 1.4 节)结束时产生下一到达间隔时间，代表了另一个依然有效却非同步的代码；同样，看不到任何好处。

表 11.4 $M/M/1$ 与 $M/M/2$ 对比：正确(代码 1)和不正确(代码 2、代码 3)同步 CRN 的统计结果

	代码 1	代码 2	代码 3
$S^2(100)$	0.07	16.80	12.00
90%置信区间半长	0.04	0.67	0.57
$\hat{\rho}$	0.07	0.43	0.42
$\widehat{\mathrm{cor}}(X_{1j}, X_{2j})$	0.997	0.02	−0.03

◀

本章到目前为止的例子(除例 11.2)都是围绕着 $M/M/1$ 与 $M/M/2$ 配置的比较展开的。对这些非常简单的系统实现恰当同步 CRN，相对说来是容易的。它们彼此也相当类似，大概通过 CRN 得到的方差减小是戏剧性的，我们也已经看到。下面的两个例子所包含的模型明显地更为复杂，也看做将 CRN 用于稳态参数(参见例 9.3 和例 9.5)。

例 11.5 在第 2.7 节的制造系统模型中，回想我们在第 2.7.3 小节最后的讨论，正如在表 2.1 的最后三行中那样，令配置 1，配置 2 和配置 3 分别为在站 1，站 2 和站 4 处各增加一台机器的制造系统。并且对于 $i=1$，2，3，令 ν_i 是在配置 i 中的稳态期望总平均作业总延误时间。假设对三个配置的每个进行 100 个 8 小时工作日的仿真，但用这 100 天的前 10 天作为预热期且只对后 90 天搜集数据，我们希望估计 $\zeta_{12}=\nu_1-\nu_2$，$\zeta_{13}=\nu_1-\nu_3$ 与 $\zeta_{23}=\nu_2-\nu_3$(参见习题 2.7)。令 X_{ij} 作为第 j 次重复运行的观测值是配置 i 的这 90 天的总平均作业总延误时间，并令 $Z_{12j}=X_{1j}-X_{2j}$，$Z_{13j}=X_{1j}-X_{3j}$，以及 $Z_{23j}=X_{2j}-X_{3j}$。我们假设 10

天的预热周期是足够的，那么 $E(Z_{i_1 i_2 j}) \approx \zeta_{i_1 i_2}$。

我们进行了不同配置完全彼此独立的运行(在表 11.5 中标以"I")，之后，在三种配置间使用了 CRN。对于每一种配置，我们使用相同的作业到达间隔时间并使作业类型顺序相同。此外，在一项作业到达系统并且确定其类型后，我们马上产生其服务需求并将它们作为该作业的附加属性存储起来，此后当它沿着其路径移动通过系统时将需要这些需求。因此，当一项作业在一个特定的站进入服务时，它的服务时间取自它在该站的适当的属性。用这种方式，我们对所有的随机源使用同步的 CRN。还要注意，在这个例子中，所有类型的随机变量都是在仿真时间的同一时刻产生的(一个作业到达时)，所以，单一随机数流可以用于所有随机源。

表 11.5 三种制造系统配置的 CRN

	I	CRN	方差减小(%)
$S^2_{12}(10)$	6.27	1.46	77
$S^2_{13}(10)$	10.40	2.35	77
$S^2_{23}(10)$	23.08	0.87	96

◀

令 $S^2_{i_1 i_2}(10)$ 是 $\mathrm{var}(Z_{i_1 i_2 j})$ 的常规的无偏估计，我们由使用独立采样和 CRN 两种方法进行 10 次独立重复运行中计算出来的，如表 11.5 给出的。这里，根据被比较的两种配置的不同，CRN 使得方差减小范围从 77% 到 96%。在达到所要求的置信区间半长所需要的重复运行次数方面，我们可以像在例 11.3 中来计算，为比较配置 1 和配置 2，独立采样需要进行的重复运行次数是 CRN 的 4.29(=6.27/1.46)倍，对于配置 1 和配置 3，是 4.43 倍，如果我们对配置 2 和配置 3 之间的差别感兴趣，则大于 26 倍。

例 11.6 在例 10.5 中，我们比较了来自例 9.25 的制造设备的两种配置。在第二种配置中，平均检测时间小一些。我们再次对工件在系统中的稳态平均时间感兴趣，因此我们使用重复运行/删除法(第 9.5.2 小节)，并对两种配置做滑动平均图，如在图 10.3 所示的那样。再次令 l_i 和 m_i 为配置 i 的预热周期长度(按零件计算)和最小重复运行长度，我们得到，$m_1=m_2=9\,945$，$l_1=l_2=1\,796$。正如在例 10.5 中那样，我们进行 $n=20$ 次独立对的运行，但我们现在使用各自的流以使本模型中的所有六个随机源的随机数均达到同步[到达间隔时间、机器加工时间、检测时间、好/坏决策、机器运转时间(上线时间)、机器维修时间(下线时间)]。再一次，我们由这 20 次独立重复运行构建系统中稳态平均时间之间差值的置信水平为 90% 的置信区间，并得到了置信区间为 1.98±0.18；回忆一下，在例 10.5 中相应的区间为 2.36±0.31，每种配置 20 次重复运行，有接近相同的运行长度和接近相同的初始数据被删除。因此，CRN 使得性能度量差的置信区间大小比原来的减小了 42%，相应的估计方差减小了 66%。作为 CRN 正确工作的一个更直接的证据，我们估计得到 X_{1j} 与 X_{2j} 间的相关性为 0.98。◀

下面的 CRN 的一对例子是关于库存模型的，对于库存模型来说，很难达到对所有随机源的完全正确同步；因此，我们将使用适当同步 CRN 来产生某些输入，而其他则独立产生。

例 11.7 对于(在第 1.5 节中定义的)库存模型，我们在例 10.3 中讨论过，回忆一下，对于配置 1，有 $(s, S)=(20, 40)$，而对于配置 2，有 $(s, S)=(20, 80)$。对于每一种配置我们可以使相同大小的需求量安排在相同的时刻发生，也就是说，我们对需求量和需求间隔时间两个随机源应用 CRN(使用各自的流)。但是，由于 S 值不同，对两种策略来说，订货发生的时间和数量一般是不同的，从而，在两种策略下订货发生的次数也是不同的。因此，不清楚我们如何才能将交货延迟随机变量进行合理的匹配(或者甚至将它们匹配是否有意义也不清楚)，所以我们只在不同配置中独立地产生它们(在例 11.2 中我们尝试过

实现交货延迟的同步，以表明在第 2、3 次重复运行中随机数是如何脱离同步的)。

像在例 10.3 中那样，我们进行 $n=5$ 次独立仿真对，但这里使用部分 CRN，正如刚才说明的。我们得到 $\overline{Z}(5)=3.95$，以及 $\mathrm{var}[\overline{Z}(5)]=0.27$，因此双 t 90% 置信区间为[2.84，5.06]。将其与例 10.3 中独立采样的结果相比，估计方差减小了大约 89%，置信区间的半长减小了大约 67%。因此，在独立采样下每个系统大概要进行 45 次重复运行得到的置信区间才与部分 CRN 只进行 5 次独立重复运行所能得到的一样小。◀

下面的例子说明第 10.3 节中讲到的多置信区间方法可以使用 CRN 进行锐化。

例 11.8 现在考虑表 10.4 中定义的 5 种不同策略配置。与在例 10.7 中一样，我们首先关注策略 1，其中$(s, S)=(20, 40)$，将其作为与其他 4 种策略进行比较的标准。我们重新进行例 10.7 中的分析，还是对每种策略配置进行 $n=5$ 次重复运行，但此次我们在所有 5 种策略间使用例 11.7 中说明的部分 CRN 采样计划。要求总置信水平至少为 90%，我们使用邦费罗尼不等式(参见第 10.3 节)完全如在例 10.7 中说明的那样来构建 $\mu_i-\mu_1$ 的 4 个每个置信水平为 97.5% 的置信区间；但是，这里 CRN 隐含 5 种策略的一次给定的重复运行(第 j 次)的结果不是独立的，排除了使用韦尔奇法(Welch Approach)来构建置信区间。所以，对应于表 10.6，表 11.6 只包含双 t 置信区间。但这里观察到的半长度都略小于表 10.6。此外，若仅比较双 t 方法，相比于例 10.7，CRN 能让我们辨识出一个统计上更为显著的差异(策略 2 与标准策略之间)。

表 11.6 应用 CRN 的所有与标准策略比较的单个 97.5% 置信区间(* 为显著差异)

i	$\overline{X}_i-\overline{X}_1$	双 t	
		半长	区间
2	−3.95	1.83	(−5.78，−2.12) *
3	1.02	2.65	(−1.63，3.68)
4	5.94	1.41	(4.53，7.35) *
5	19.69	2.28	(17.41，21.97) *

◀

我们还可以应用 CRN 使第 10.3.2 小节的所有两两比较分析结果更有效；在例 10.8 中使用独立采样已经做过了，其结果在表 10.7 中。由于现在有 10 个单个的置信区间，我们使每一置信区间达到 99% 水平，以使整体置信水平至少 90%。表 11.7 中的 CRN 得到的区间(同样只对双 t 方法有效)再一次表明，除($i_2=5$，$i_1=2$)一种情形外，在所有情形下 CRN 显著地减小了置信区间的长度。在此情况中表 10.7 的独立采样下得到的双-t 区间(22.12±3.80)，要比表 11.7 的 CRN 区间(23.64±5.02)小。再看表 10.7，此情况下的置信区间是观察到的最小的，这可能是由于简单采样波动造成的，并且在任何速率下两处的差值都很显著。可能更重要的是，我们在表 11.7 中看到，10 个基于 CRN 的区间有 8 个是不过 0 的(表明相应配置间的差异统计显著)，而在表 10.7 中的 10 个只有 6 或 7 个不过 0。因此，在不增加采样的情况下能锐化我们对这些策略的比较。

表 11.7 使用 CRN 的两两比较($i_1<i_2$，$\mu_{i_2}-\mu_{i_1}$)的单个 99% 置信区间(* 表示显著差异)

		双-t			
		i_2			
		2	3	4	5
i_1	1	−3.95±2.41 *	1.02±3.49	5.94±1.86 *	19.69±3.00 *
	2		4.97±5.62	9.89±3.68 *	23.64±5.02 *
	3			4.92±2.39 *	18.67±1.69 *
	4				13.75±2.03 *

例 11.9 再次考虑例 10.6 的通信网络，比较原始路由策略和建议路由策略的问题。我们再次对每一策略进行 5 次独立重复运行，长度 m=65 秒，并使用 $l=5$ 秒的预热期，但现在在两种策略间使用部分 CRN。对于每一种配置，我们都在相同时刻产生某一特定 SP 的发出报文，这报文有相同的大小及目的地。但是，当两条链路的每一条都必须以 0.5 的概率选择时，我们对两种配置使用不同的随机数。我们用式(10.1)得到[0.05，0.12](单位为毫秒)为 $\nu_1-\nu_2$ 的近似 90%的置信区间。由于置信区间是不含 0 的，两个稳态均值间的差值是统计显著的，而在例 10.6 中并不是。此外，Z_j 的样本方差减小了 97%，置信区间的半长度减小了 83%左右，X_{1j} 和 X_{2j} 间估计的相关系数为 0.94。◀

总之，我们发现，在我们已经完成的许多实际仿真研究中，我们至少可以实现不同系统配置之间的部分同步。

11.3　对偶变量法

作为本章中 VRT 的剩余内容，对偶变量(antithetic variate，AV)法是一种可应用于单一系统仿真的 VRT。如在 CRN 中一样，我们试图在各自的运行之间引入相关性，但现在我们寻找负相关性。

AV 法的核心思想至少可追溯到 Hammersley 和 Morton(1956)的蒙特卡罗仿真的论述，它对模型进行成对运行，使得对一对运行的一个的“小”的观测结果将有被另一个的“大”观测结果补偿的趋向，即两个观测是负相关的。这样如果我们成对使用两次观测结果的均值作为分析的基础数据点，与如果两个成对的观测值是独立的相比，它将趋向于更接近观测值共同期望值 μ(这是我们要估计的)。

在 AV 法的最简单形式中，通过使用互补的随机数来驱动成对的两次运行，它试图引入负相关。也就是说，如果 U_k 是在第一个运行中用于特定目标(例如产生第 i 个服务时间)的特定随机数，则我们在第二个运行中将 $1-U_k$ 用于该同一目标。使用 $1-U_k$ 代替只直接由随机数发生器产生是有根据的，因为 $U\sim U(0,1)$ 意味着也有 $1-U\sim U(0,1)$。

重要的一点是，必须保持一次重复运行使用 U_k 与在该对重复运行中使用它的补 $1-U_k$ 同步，即用于相同的目的；否则 AV 法的好处将会失去，甚至可能事与愿违。举例来说，如果 U_k 出现大值，同时经过(精确的)反变换方法用于产生一个服务时间，这会导致一个大的服务时间，效果是在该对的第一次运行中增加队列的拥塞。在这种情况下，$1-U_k$ 就会小，这样如果错误地将它用于第二次运行中产生一个到达间隔时间，该到达间隔时间就小，在第二次运行中也会增加拥塞，这与我们期望产生的效果正好相反。在第 11.2 节中提到的保持随机数同步的大多数编程技巧也都可以在这里使用，例如随机数流专用、尽可能使用反变换方法产生随机变数、有策略的舍去一些随机数、预产生随机数、在多个重复运行中预定流号等。此外，如果证明全同步太困难或似乎没有办法对所有输入使用 AV 法，我们可考虑“部分”AV 法，即在一对内对偶地产生某些输入，而其他则独立产生(参见下面的例 11.12)。那么，的确，只遍历仿真代码并用“$1-U$”替代“U”是不够的。

正如利用 CRN 一样，AV 法也有数学基础。假设我们进行了 n 对仿真运行，得到观测结果$(X_1^{(1)}, X_1^{(2)}), \cdots, (X_n^{(1)}, X_n^{(2)})$，其中 $X_j^{(1)}$ 来自第 j 对的第一次运行(只使用 U)，而 $X_j^{(2)}$ 来自第 j 对的对偶运行(使用“$1-U$”，恰当同步)。此时 $X_j^{(1)}$ 和 $X_j^{(2)}$ 都是该仿真模型的合理观测值，因此 $E(X_j^{(1)})=E(X_j^{(2)})=\mu$。而且，每对都独立于其他每对；即对于 $j_1\neq j_2$，无论 l_1 和 l_2 是否相等，$X_{j_1}^{(l_1)}$ 与 $X_{j_2}^{(l_2)}$ 是独立的(请注意，总的重复次数因此为 $2n$)。对于 $j=1,2,\cdots,n$，令 $X_j=(X_j^{(1)}+X_j^{(2)})/2$，且令 X_j 的平均值 $\overline{X}(n)$ 是 $\mu=E(X_j^{(l)})=E(X_j)=E[\overline{X}(n)]$的(无偏)点估计。由于 X_j 是独立同分布的，那么，

$$\operatorname{var}[\overline{X}(n)]=\frac{\operatorname{var}(X_j)}{n}=\frac{\operatorname{var}(X_j^{(1)})+\operatorname{var}(X_j^{(2)})+2\operatorname{cov}(X_j^{(1)},X_j^{(2)})}{4n}$$

如果一对内的两次运行是独立的，则 $\operatorname{cov}(X_j^{(1)}, X_j^{(2)})=0$。另一方面，如果我们的确

能在 $X_j^{(1)}$ 和 $X_j^{(2)}$ 间引入负相关性，则 $\mathrm{cov}(X_j^{(1)}, X_j^{(2)})<0$，这减小了 $\mathrm{var}[\overline{X}(n)]$；这就是 AV 法的目标。

然而，AV 法和 CRN 共有的另一个特点是我们不能完全确定它是有效的，以及较之 CRN，它的可行性和效率更加依赖于模型。但是，在某些情形下，已经解析地证明了 AV 法可以降低方差，虽然降低的幅度是未知的；参见 Andréasson(1972)、George(1977)、Hammersley 和 Handscomb(1964)、Mitchell(1973)、Wilson(1979)、Rubinstein，Samorodnitsky 和 Shaked(1985)、Bratley，Fox 和 Schrage(1987，第 2 章)。一般来说，我们不可能事先知道方差减小会达到多大。对判断在特定情形下 AV 法是否是个好主意来说，进行类似讨论 CRN 那样的试验性研究可能是有用的。

一个模型应满足 AV 法运行的基本要求是：模型对用于特定目的随机数的响应在每个方向上都是单调的。举例说来，在大多数排队模型中，用于产生服务时间的一个大随机数(通过精确的反变换方法)将产生大服务时间，并导致拥塞加剧；因此，我们会希望拥塞度量对用于产生服务时间的随机数的响应应是单调递增的。如果一个模型对一个小的 U_k 响应大，对接近 0.5 的 U_k 就小一些，然后对大的 U_k 响应又变大(参见习题 11.3)，AV 法可能会发生反效果。我们力劝分析人员提供某一类 AV 法会管用的证据，或者从模型结构的“物理”性质来证明，或者通过初步试验。如应用 CRN 一样，建议使用反变换法来产生模型的输入变量，通过至少在随机变量产生这个中间环节保证单调性来提高 AV 法所需要的单调性。Franta(1975)给出了如果使用其他方法导致失败的例子，而 Schmeiser 和 Kachitvichyanukul(1990)给出了一种快速非反变换法，该方法在随机变量产生这个中间环节确实导致所要求的负相关。

例 11.10 考虑 $\rho=0.9$ 的 $M/M/1$ 队列，如在例 11.1、例 11.3 和例 11.4 中那样，那么，现在“观测值” $X_j^{(l)}$ 是 100 个顾客延迟时间的平均。从模型的结构，假设大的到达间隔时间趋向于使 $X_j^{(l)}$ 变小(反之亦然)，大的服务时间一般得到较大的 $X_j^{(l)}$。此外，如果我们使用第 8.3.2 小节中的方法产生指数到达间隔时间和服务时间变量，我们可以期望 AV 管用。我们使用在每对运行中独立采样[那么，$\mathrm{cov}(X_j^{(1)}, X_j^{(2)})=0$]，以及 AV 法，进行了 $n=100$ 对独立运行，通过专用各自的随机数流产生到达间隔时间和服务时间来保证同步。结果如表 11.8 所示，并说明 AV 法减小了 X_j 的估计方差 $S^2(100)$ 的 60%。如果我们使用 100 个 X_j 来构造 μ 的近似 90% 置信区间，则在独立采样下其半长度为 0.36，而在 AV 法下其半长度降为 0.23，降低了 36%。注意，在 AV 法的情况下，$X_j^{(1)}$ 与 $X_j^{(2)}$ 间的估计的相关系数为 −0.52，证实了 AV 法引入了所希望的负相关性。注意，这里 AV 法在独立采样基础上的额外花费是可以忽略不计的，因为 AV 法几乎不需要额外的编程，而只需做 1 减去随机数的减法；两种方法都是需要 200 次的独立仿真运行。

表 11.8 在 $M/M/1$ 队列情形 AV 法的统计结果

	独立	AV
$S^2(100)$	4.84	1.94
90%置信区间半长	0.36	0.23
$\widehat{\mathrm{cor}}(X_j^{(1)}, X_j^{(2)})$	−0.07	−0.52

图 11.6a 表示了对于每个 j 的独立采样的 $X_j^{(1)}$ 单个值(实心圆)和 $X_j^{(2)}$ 的单个值(空心圆)。粗实线将各个 j 的 X_j 值连接起来。图 11.6b 所示的与图 11.6a 所示的类似，不过其中使用了 AV 法，我们看到在图 11.6b 所示的粗实线的抖动程度稍微小一些，表明在 AV 法情况下 X_j 的方差较小。在图 11.6b 中我们还能看到这样一个趋势，即在 AV 法下一个在虚线(高度为 μ)一侧的 $X_j^{(1)}$ 被在虚线另一侧的一个 $X_j^{(2)}$ 补偿，而在独立采样的图 11.6a 中比较小。AV 法下的 X_j 的变化程度较低，在图 11.7 中更明显地得到证实，这里显然 X_j

的分散度更窄了。最后，图 11.8给出了在独立采样(空心三角)与 AV 法(实心三角)两种情形下($X_j^{(1)}$，$X_j^{(2)}$)对的相关性图。在 AV 法情形的确出现了一些负相关，因为实心三角表现出了某种向下倾斜的倾向。

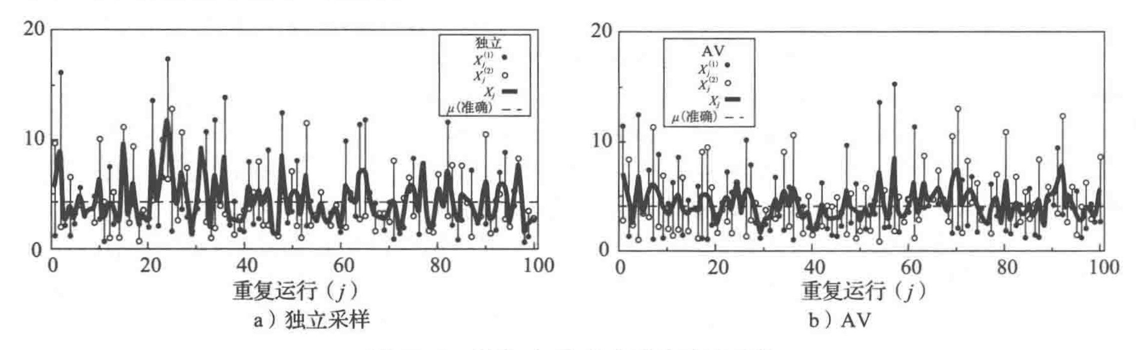


图 11.6 $M/M/1$ 队列的单个重复运行

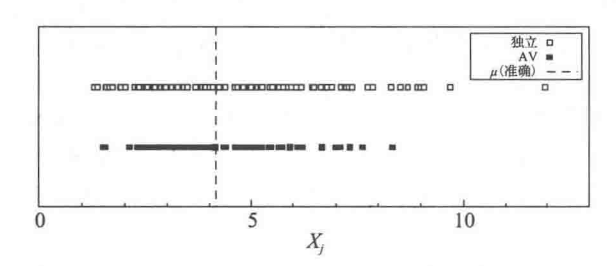


图 11.7 使用独立采样和 AV 法的 $M/M/1$ 队列的对内均值

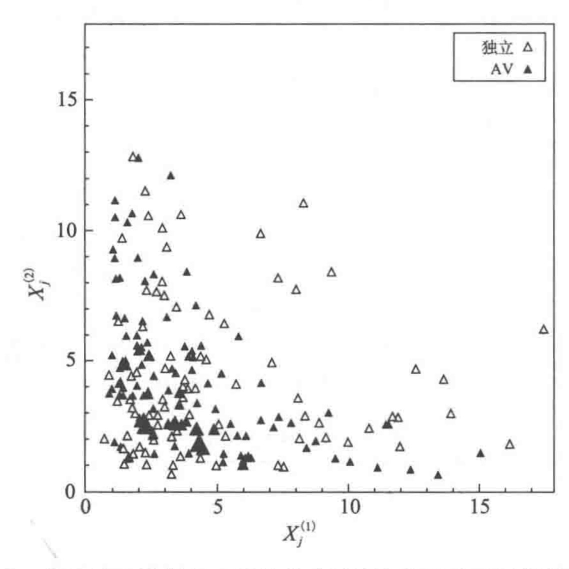


图 11.8 使用独立采样和 AV 法的 $M/M/1$ 队列成对运行的相关性 ◀

在例 11.10 中通过 AV 法引入的相关性的幅度没有我们在例 11.3 中观察到的使用 CRN 比较 $M/M/1$ 和 $M/M/2$ 队列所引入的相关性那么强，因此 AV 法所得到的方差减小较之通过 CRN 得到的要小。这是 VRT 的成功通常如何依赖于模型特性的一个很好的说明。在例 11.3中，两种配置(在完全同步 CRN 下，称为 “A&S”)的输入变量的唯一差异是 $M/M/2$ 队列的服务时间在每种情形下恰好为 $M/M/1$ 队列的 2 倍，因而两个服务时间是线性相关的(参见习题 4.11)。但是，在例 11.10 的 AV 法方案中，成对运行第一次和第二次的输入变数不是线性相关的，基本上归结为 $\ln U$ 与 $\ln(1-U)$的关系，其中 U 是随机

数，正如在第8.3.2小节中所讲述的那样，因为指数随机变量的产生是对随机数取自然对数。在CRN和AV法的两个例子中，输入变数后来都会被仿真模型本身进行非线性变换。重新回忆一下，协方差和相关系数包含CRN和AV法两者减小方差的数量，它们只是线性关系的度量，我们看到，在例11.3的CRN应用中线性度较大，与例11.10的线性度较小的AV法应用相比，显然其方差减小更多。

例11.11 在AV法应用到排队模型的特定情形下，Page(1965)建议过另一种类型的对偶采样，即在一对运行的第二次中不使用$1-U$代替U。由于排队系统的性能度量一般来说对大的到达间隔时间的反应和它对大的服务时间的反应相反，Page就建议，在一对运行的第一次中用于产生到达间隔时间的随机数，而在第二次运行中改为用于产生服务时间，并反过来也一样。我们将这一思想在例11.10的$M/M/1$模型上加以实现，得到了65%的方差减小。这种交换使用随机数的一般方法也应适用于其他类型的模型中引入负相关性。 ◀

我们最后一个AV法的例子说明其在非排队模型中应用，同时对该例来说，只有部分同步是有意义的。

例11.12 为了看到AV法的响应单调性和潜在成功的物理原理，虽然需要做某些额外的工作(参见习题11.5)，但还是可以将AV法应用于第1.5节的库存模型。取$(s, S)=(20, 40)$，假设参数以及输出变量均与例11.7中的相同。也像例11.7中的做法一样，在一对的两次运行间，我们将只对需求间隔时间和需求量上应用AV法，并独立地产生交货延迟。我们再次做$n=100$对独立运行，首先使$X_j^{(1)}$和$X_j^{(2)}$独立，并得到$\mathrm{var}(X_j)$的估计为8.55，而在AV法下可比较的方差估计为3.26，降低了约62%。因此，我们看到仅仅部分同步的AV法也仍然得到值得一做的方差减小。 ◀

由于CRN与AV法之间的相似性，在比较几种备选系统配置时，合理的想法是考虑将它们一起使用。对于每一种配置，各自使用AV法且在配置间使用CRN，最初我们以为会得到更大的方差减小。但是，根据最近的检查[参见文献Kleijnen(1974，第207-238页)和习题11.11]，我们发现，即使AV法和CRN工作得合适，即引入了我们所希望符号的相关性，但是一定的“交叉协方差”(特别地，配置1的对偶运行对的第一次运行与配置2的相应对偶对的第二次运行之间的协方差，反之亦然)会带着错误的符号进入相关的方差表达式，它可能会增大方差。因此，没法搞清将AV法和CRN结合起来比较多种系统配置是否是一个好想法。Schruben和Margolin(1978)，Schruben(1979)以更一般性的方式研究了在各种不同仿真实验中的方差减小的相关性引入策略中的随机数分配问题。

AV法的更一般的版本已经开发出来，包括被估计的量(除了期望之外)以及引入负相关的方法(除了采用互补随机数之外)；细节请参见Cheng(1982，1984)、Fishman和Huang(1983)、Wilson(1983)。Nelson(1990b)给出了AV法的一个潜在的副效果，他证明了，在存在初始化偏差时，AV法(与控制变量法结合，在第11.4节中讨论)可以改进点估计和区间估计两者的性能。Avramidis和Wilson(1998)使用AV法改进了分位数的估计(参见第9.4.2小节)。

11.4 控制变量法

类似于CRN和AV法，控制变量(control variate，CV)法力图利用某些随机变量间的相关性来获得方差的减小。根据使用的CV法技术种类不同，这种相关性在仿真过程中可能会自然产生或在辅助仿真中使用CRN引入。

在原理上，至少有一种对CV法的直观表述。令X是一个输出随机变量，例如是队列中初始100个顾客的延误时间的平均值，并假设我们要估计$\mu=E(X)$。假定Y是仿真中所包括的另一个随机变量，它被认为与X相关(无论正相关或负相关)，且我们已知$\nu=E(Y)$的值。举例来说，Y可能是上述排队模型中完成了服务的前99名顾客的服务时间的

平均值，那么，由于我们从某种已知的输入分布产生了服务时间变量，我们应该知道 Y 的期望值(习题 11.12 论述了关于 Y 的精确定义的细节问题)。一个合理的猜想是，大于均值的服务时间(即 $Y>\nu$)将导致大于均值的延误时间(即 $X>\mu$)，反之亦然；即 Y 和 X 是相关的，在此种情况下是正相关的。因此如果我们运行仿真并发现 $Y>\nu$(由于我们已知 ν，所以我们肯定能知道这一点)，我们便会猜想 X 也大于其期望 μ(虽然我们并不能确知这一点，除非 Y 和 X 之间的相关性非常好)，相应地我们将 X 下调一点。相反，如果得到 $Y<\nu$，则我们也会猜想 $X<\mu$，那么将 X 上调一些。用这种方法，我们应用我们的有关 Y 的期望的知识来调节 X(向下或向上)，以靠近其期望 μ，这样，减少 X 从一次运行到下一次运行在 μ 周围的变化。我们称 Y 为 X 的控制变量，因为它用于调节 X，或者部分地“控制”它。

与 CRN 和 AV 法不同，CV 法的成功并不依赖于相关性的特定符号。如果 Y 和 X 是负相关的，我们将其想象为，如果 Y 是在前一节的例子中的前 100 个产生的到达间隔时间的均值(很分散的到达会产生较低的拥塞水平)，如果 $Y<\nu$，我们只要简单地上调 X，而如果$Y>\nu$，就下调。

要实现上述思想，我们必须量化上调或下调 X 的量。方便的做法是，将该量表示为偏移量 $Y-\nu$，即 Y 偏离其期望 ν 的量。令 a 是一个常数(下面来确定)，它的符号与 Y 和 X 间的相关系数的符号相同。在此前的例子中，X 是平均排队延误时间，而 Y 是平均服务时间，因此 a 应该是某个正数。我们使用 a 来改变(增大或缩小)偏离量 $Y-\nu$ 的比例以达到对 X 的调整，从而定义“控制”估计值为：

$$X_C = X - a(Y-\nu)$$

注意，若 Y 和 X 正相关，则 $a>0$，一旦 $Y>\nu$，我们应下调 X，当 $Y<\nu$ 时，上调，正如所希望的那样；如果 Y 和 X 是负相关的，则在这种情形下 $a<0$，则反着做。

由于 $E(X)=\mu$，同时 $E(Y)=\nu$，显然，对于任意正数 a，都有 $E(X_C)=\mu$，即 X_C是 μ 的无偏估计，其方差可能比 X 的小。特别是，有

$$\text{var}(X_C) = \text{var}(X) + a^2\text{var}(Y) - 2a\text{cov}(X,Y) \tag{11.1}$$

因此，X_C是一个小于 X 的随机变量，当且仅当

$$2a\text{cov}(X,Y) > a^2\text{var}(Y)$$

上式是否成立，这主要取决于 Y 和 a 的选择。在很多 CV 法的处理中，只考虑 $a=1$[如果我们认为 $\text{cov}(X, Y)>0$]和 $a=-1$[如果我们认为 $\text{cov}(X, Y)<0$)两种特别情况，但这需要更为严格的条件，即 $|\text{cov}(X, Y)|>\text{var}(Y)/2$，才能实现方差的减小。因此，简单地设 $a=\pm1$ 会使成功的所有负担都落在 Y 的选择上；通过允许使用其他 a 值，我们可做得更好。

对于一个给定的 Y，为找到“最佳”的 a 值，我们可将式(11.1)的右侧视为 a 的函数 $g(a)$，并置它的导数为 0，即

$$\frac{\text{d}g}{\text{d}a} = 2a\text{var}(Y) - 2\text{cov}(X,Y) = 0$$

并求解出最优(方差最小化)值为：

$$a^* = \frac{\text{cov}(X,Y)}{\text{var}(Y)} \tag{11.2}$$

$\text{d}^2g/\text{d}a^2=2\text{var}(Y)$，它当然为正，是 a^* 为 $g(a)$ 最小化因子的充分条件，相反则最大化因子或拐点。式(11.2)的含义之一是，如果 Y 和 X 是强相关的，即 $|\text{cov}(X, Y)|$ 值大，则 a^* 的值也增加，从而我们将会对 X 做更大的调整，因为我们更会相信有关 Y 偏离 ν 的信息就会告诉我们 X 偏离 μ 的情况。而且，如果 Y 本身可变性较小，即 $\text{var}(Y)$小，我们也会得到较大的 a^*(和对 X 更大的调整)，因为我们在观测到的 Y 本身值的精度方面有更大的置信度。

将式(11.2)的 a^* 代入到式(11.1)的右侧，我们得到对所有 a 的选择的最小方差调正的(控制的)估计值 X_C^* 的方差为：

$$\operatorname{var}(X_C^*)=\operatorname{var}(X)-\frac{[\operatorname{cov}(X,Y)]^2}{\operatorname{var}(Y)}=(1-\rho_{XY}^2)\operatorname{var}(X)$$

其中，ρ_{XY}是 X 和 Y 之间的相关系数。

因此，利用 a 的最优值 $a*$，则最优控制的估计值 X_C^* 的变化不可能大于未控的 X，而且，若 Y 完全与 X 相关，则事实上会有更小的方差。此外，X 和 Y 之间的相关性越强，其方差减小得越多——在极端情况下，当 $\rho_{XY}\to\pm1$，在事实上我们看到 $\operatorname{var}(X_C^*)\to0$。直觉上，这就是说，如果 X 和 Y 之间的相关性接近完美(±1)，则我们可以每次都将 X 控制到几乎准确地到 μ，因此在实际上消除了它的全部方差。

虽然如此，在实际中，事情完全不是如此完美。根据控制变量 Y 的源及其自然属性，我们或许会或许不会知道 var(Y)的值，而且我们肯定不知道 cov(X，Y)，使得不可能求得 a^* 的准确值。相应地，提出了一些从仿真运行来估计 a^* 的方法，我们下面介绍其中一种较为简单的方法，是由文献 Lavenberg，Moeller 和 Welch(1982)，Lavenberg 和 Welch(1981)提出的，此方法也可用于构建 μ 的置信区间[正如所说明的，这种方法适用于简单重复运行的终止型仿真(参见第 9.3 节)，虽然通过使用第 9.5.2 小节重复运行/删除法，或通过使用批均值替代重复均值的方法，它或许可应用于稳态参数，正如在第 9.5.3 小节中讨论的]。

此方法只是使用样本估计值来代替 cov(X，Y)和 var(Y)。假设我们进行了 n 次独立重复运行以获得 X 的 n 个独立同分布观测值 X_1，X_2，…，X_n，以及 Y 的 n 个独立同分布观测值 Y_1，Y_2，…，Y_n。设 $\overline{X}(n)$和 $\overline{Y}(n)$分别是 X_j 和 Y_j 的样本均值，令 $S_Y^2(n)$是Y_j的无偏样本方差，则 X 和 Y 间的协方差可由下式估计得到(参见习题 4.25)：

$$\hat{C}_{XY}(n)=\frac{\sum_{j=1}^{n}[X_j-\overline{X}(n)][Y_j-\overline{Y}(n)]}{n-1}$$

且 a^* 的估计值则为：

$$\hat{a}^*(n)=\frac{\hat{C}_{XY}(n)}{S_Y^2(n)}$$

达到 μ 的最终点估计为：

$$\overline{X_C^*}(n)=\overline{X}(n)-\hat{a}^*(n)[\overline{Y}(n)-\nu]$$

我们必须注意到，因为用随机变量$\hat{a}^*(n)$代替了常数 a^*，而$\hat{a}^*(n)$一般来说与$\overline{Y}(n)$是不独立的(由相同的仿真输出数据来计算)，所以我们不能轻易地越过 $\overline{X}_C^*(n)$的第二项中的系数得到期望值。那么，不幸的是，不像 X_C^* 和 X_C，$\overline{X}_C^*(n)$一般是 μ 的有偏估计。Lavenberg，Moeller 和 Welch(1982)对使用此方法会带来的这一偏差的严重程度以及方差减小的数量进行了研究。

Kleijnen(1974)以及 Lavenberg，Moeller 和 Welch(1982)讨论了基于对折法以降低 $\overline{X}_C^*(n)$偏差的另一种估计 a^* 的方法。Cheng 和 Feast(1980)、Bauer(1987)研究了当我们已知控制变量的方差时的 CV 法问题。Nelson(1989，1990a)研究了在稳态仿真中使用批均值方法的 CV 法，同时研究和评估了多种处理点估计偏差问题的方法以及其相关问题；Yang 和 Nelson(1992)将这种分析扩展到多变量批均值与 CV 法合作。Avramidis 和 Wilson(1993)讨论了分割仿真输出数据来估计 a^* 的方法。

例 11.13 为巩固本节中我们非正式地讨论过的例子，令 X 为到达 $M/M/1$ 队列的前 100 名顾客在队列中的平均延误时间，该队列出发时为空且闲，平均到达间隔时间为 1 分钟，平均服务时间为 0.9 分钟；这是在例 11.1、例 11.3、例 11.4、例 11.10 和例 11.11 中我们用到的同一个模型。Y 作为 X 的控制变量，令其为 99 个服务时间的均值，这些服

务时间是完成该模型的一次重复运行所必需的；因为仿真在第 100 个服务开始时结束，它的值对输出并无影响，从而并未被包含在 Y 中。因此 Y 是一个固定个数(参见习题 11.12)的独立同分布服务时间，其期望为 0.9，也就是 $\nu=E(Y)=0.9$。

我们进行 $n=10$ 次独立重复运行，并观察到在表 11.9 中给出的 10 个 X_j 和 10 个相应的 Y_j。由这些数据得到，$\overline{X}(10)=3.78$ 与 $\overline{Y}(10)=0.89$；因此，平均服务时间略低于其期望值 $\nu=0.9$，且队列中的平均延误时间也比其期望 $\mu=4.13$ 低(实际上在该人为的例子中我们实际上知道其期望值)，正如在本节开头所建议的那样。此外，我们可以得到 $S_X^2(10)=13.33$，$S_Y^2(10)=0.002$，$\hat{C}_{XY}(10)=0.07$，就有 X 和 Y 之间的相关系数为 0.43，证实了我们关于 X 和 Y 之间应是正相关的感觉。最后，我们得到 $\hat{a}^*(10)=35.00$，则 $\overline{X}_C^*(10)=4.13$，这的确比未控制的估计 $\overline{X}(10)=3.78$ 更接近 μ。

表 11.9 对 $M/M/1$ 队列使用 CV 法的平均延误时间(X_j)和平均服务时间(Y_j)

j	X_j	Y_j
1	13.84	0.92
2	3.18	0.95
3	2.26	0.88
4	2.76	0.89
5	4.33	0.93
6	1.35	0.81
7	1.82	0.84
8	3.01	0.92
9	1.68	0.85
10	3.60	0.88

为了看看 $\text{var}[\overline{X}_C^*(10)]$是否实际上小于 $\text{var}[\overline{X}(10)]$，我们重复前一节的 10 次重复运行实验 100 次，得到了 100 个 $\overline{X}_C^*(10)$和 $\overline{X}(10)$的独立观测值。从这些数据中我们得到 $\text{var}[\overline{X}(10)]$的估计为 0.99，而我们的 $\text{var}[\overline{X}_C^*(10)]$的估计为 0.66，降低了三分之一。$\overline{X}(10)$和 $\overline{Y}(10)$间的相关性估计值为 0.67，它是正相关，正如我们预料的那样。同时，我们从 100 个观测值对 $\overline{X}_C^*(10)$进行估计，$E[\overline{X}_C^*(10)]$的 95%置信区间为 4.18 ± 0.16，它包含 μ，表明，无论如何，在此例中使用 $\hat{a}^*(10)$代替 a^*，在控制的估计值中所引入的偏差并不明显。◀

在例 11.13 中我们选择了服务时间的均值作为我们的控制变量，但是，正如下面两个例子表明的那样，选择什么作为控制变量绝非是显而易见的问题。

例 11.14 我们重做例 11.13 中的实验，但此次我们使用前 100 个到达间隔时间的均值作为控制变量 Y。我们可以确定，实验中将至少总有这么多个到达间隔时间，因此我们基于所产生的到达间隔时间得到的控制变量是固定数量的(参见习题 11.12)，所以我们知道 $E(Y)=1$。如例 11.13 中说明的一样，进行一组 100 次的实验每次重复运行的 $n=10$，我们可以基于该控制变量估计 $\text{var}[\overline{X}_C^*(10)]$为 0.89，与 $\overline{X}(10)$的估计方差 0.99 相比只降低了 10%。因此，看来我们原来选择服务时间的均值作为控制变量比利用平均到达间隔时间的想法更好。◀

例 11.15 我们能否以某种方式使用两个？令 $Y^{(1)}$ 为例 11.13 中使用的平均服务时间控制变量，$Y^{(2)}$ 为例 11.14 中使用的平均到达间隔时间控制变量。那么，由于 $\text{cov}(X, Y^{(1)})$和 $\text{cov}(X, Y^{(2)})$的正负号恐怕是相反的，若我们定义一个新的控制变量 $Y=Y^{(1)}-Y^{(2)}$，我们也许可以将两者的信息组合起来；我们期望，在 $Y^{(1)}$ 和 $Y^{(2)}$ 两个支持下的 $\text{cov}(X, Y)>0$。的确，当在每次重复运行 $n=10$ 的 100 次新实验中使用新的方案时，$\overline{X}(10)$和 $\overline{Y}(10)$间的估计相关系数为 0.77，而控制估计值的方差的估计为 0.56，方差的减

小从未加控制的估计值0.99降低了43%，且比单独使用$Y^{(1)}$或$Y^{(2)}$都好。然而，在这种情况下明显地引入了点估计偏差，譬如$E[\overline{X}_C^*(10)]$的95%置信区间为4.38±0.15，它并不包含$\mu=4.13$；这是否可行取决于人们如何选择对点估计的偏差与方差之间的权衡。◀

在例11.15中实际有两个不同的控制变量，我们能以一种合理的方式将其组合来得到单一控制变量。但是，为什么我们要将它们相减而不是一个变量除以其他的变量？或者，为什么不令$Y=Y^{(1)}-2Y^{(2)}$来替代？在复杂模型中，可以有很多潜在的控制变量可用，同时可能很难建议一种最佳办法来将它们全部揉成一个。此外，就算我们能以一种合理的方式组合它们，我们也许并未将其信息用得最好。在例11.15中，我们令该控制变量为$Y=Y^{(1)}-Y^{(2)}$，这样，控制估计为：

$$X_C = X - a(Y-\nu) = X - a(Y^{(1)}-\nu^{(1)}) - a(-Y^{(2)}+\nu^{(2)})$$

其中，$\nu^{(l)}=E(Y^{(l)})$。那么，在这个式子中，我们强制每一个控制变量($Y^{(1)}$和$-Y^{(2)}$)用相同的系数a进入调整，这也许不是最好地使用它们的信息。一个符合逻辑的改进是允许两个控制变量拥有不同的权重，并定义

$$X_C = X - a_1(Y^{(1)}-\nu^{(1)}) - a_2(Y^{(2)}-\nu^{(2)})$$

然后我们可以求(或估计)出使$\mathrm{var}(X_C)$的最小化的权重a_1和a_2，正如我们前面只有单个控制变量时做的那样。

这个思想易于一般化为这样的情况，其中我们有m个控制变量$Y^{(1)}$，$Y^{(2)}$，…，$Y^{(m)}$，分别具有已知的期望值$\nu^{(1)}$，$\nu^{(2)}$，…，$\nu^{(m)}$。一般(线性)控制估计器为：

$$X_C = X - \sum_{l=1}^{m} a_1(Y^{(1)}-\nu^{(1)})$$

其中，a_l是待确定(和估计)的实数。

我们不仅允许X和控制变量之间有相关性，也允许控制变量本身之间有相关性，得到

$$\mathrm{var}(X_C) = \mathrm{var}(X) + \sum_{l=1}^{m} a_l^2\,\mathrm{var}(Y^{(l)}) - 2\sum_{l=1}^{m} a_l\,\mathrm{cov}(X,Y^{(l)})$$
$$+ 2\sum_{l_1=2}^{m}\sum_{l_2=1}^{l_1-1} a_{l_1}a_{l_2}\,\mathrm{cov}(Y^{(l_1)},Y^{(l_2)}) \qquad (11.3)$$

求式(11.3)右面对于每一个a_l的偏导数并让偏导数等于零，得到一个组m个线性方程，以求解m个最小化方差的权值(参见习题11.13)。像单控制变量的情形一样，这些最优权重值也需要进行估计，因此在控制估计器中也可能会引入偏差。Lavenberg，Welch和Nelson(1981)讨论了这一问题及相关问题。最优权重的估计被证明与在某种线性回归模型中的系数的最小二乘法估计是相同的，因此CV法有时也称为回归采样法。

我们用简要讨论控制变量的寻找与选择来结束本节。正如我们已经看到的，一个好的控制变量应该与输出随机变量X强相关，以便为我们提供更多的有关X的信息并使X调整得好。我们也希望控制变量本身方差小。可通过分析模型的结构，或通过初步实验来推进这样的控制变量的寻找。根据这些既定目标，提出如下三类控制变量的通用源。

- **内部的**：输入随机变量或它们的简单函数(例如均值)常用作控制变量。在例11.13到例11.15中使用的所有控制变量都是内部的。它们的期望一般说来都是已知的(参见习题11.12的注意事项)，且简单分析它们在模型中的作用便可以得出它们将如何与输出随机变量相关。最重要的是，为使仿真得以运行，内部控制变量无论如何是必须要产生的，因此它们基本上不会增加系统的开销；因此，即便它们并不能大大地减小方差，它们也是值得使用的(参见习题11.1关于VRT效率的经济模型)。关于各类内部CV法应用的详细讨论可在以下文献中找到，它们是Iglehart和Lewis(1979)、Lavenberg，Moeller和Sauer(1979)、Lavenberg，Moeller和Welch(1982)、Wilson和Pritsker(1984a，1984b)。

- **外部的**：也许，我们进行仿真是因为我们无法解析求得 $\mu=E(X)$的值。虽然如此，大概如果我们进行一些附加的简化假设来修改模型，我们或许就能计算出简化模型的输出随机变量 Y 的期望 ν。虽然我们并不愿意在我们的实际模型中进行这些简化假设，因为它们可能实际上损害了模型的有效性，Y 可能用作 X 的控制变量。这样，我们也许使用 CRN 与实际模型一起对简化模型进行仿真(第 11.2 节)，并希望 Y 与 X 相关，也许是正相关。不像内部 CV 法，本方法不是无代价的，因为它包括第二个仿真以得到控制变量；因此，外部 CV 法的 Y 与 X 间的相关性必须要强于如果 Y 是内部 CV 法情形以得到补偿。外部 CV 法的例子可在下述文献中找到，它们是 Burt，Gaver 和 Perlas(1970)、Gaver 和 Shedler(1971)、Gaver 和 Thompson(1973)、Schmeiser 和 Taaffe(1994)、Nelson(1997)、Irish(2003)，以及习题 11.14。
- **使用多个估计器**：在某些情形下，我们可能有多个 μ 的无偏估计器 $X^{(1)}$，$X^{(2)}$，…，$X^{(k)}$，其中，$X^{(i)}$之间可以是独立的或是不独立的。举例来说，我们可能使用将在第 11.5 节中介绍的间接估计法会发生这种情况。若 b_1，…，b_k是任意实数(不需要一定为正数)，并且其和为 1，则

$$X_C=\sum_{i=1}^{k}b_iX^{(i)}$$

也是 μ 的无偏估计。由于 $b_1=1-\sum_{i=2}^{k}b_i$，我们可以 X_C表达为：

$$X_C=(1-\sum_{i=2}^{k}b_i)X^{(1)}+\sum_{i=2}^{k}b_iX^{(i)}=X^{(1)}-\sum_{i=2}^{k}b_i(X^{(1)}-X^{(i)})$$

因此，对于 $i=2$，3，…，k，我们可以将 $Y_i=X^{(1)}-X^{(i)}$看做 $X^{(1)}$的 $k-1$ 个控制变量。

由上可以看出，对复杂模型来说，可能的控制变量数目可以非常大。但是，将全部使用它们并不一定是个好想法，这是因为它们可带来的方差减小伴随着与需要估计最优的 a_l 有关的方差增加。Bauer 和 Wilson(1992)提出了一种在可用的控制变量中选择最优子集的方法，其假设是我们已知哪些与方差及协方差有关。也可参见文献 Rubinstein 和 Marcus (1985)、Venkatraman 和 Wilson(1986)、Porta Nova 和 Wilson(1993)。

我们关于控制变量的讨论集中在均值估计上。关于使用 CV 法来锐化概率和分位数估计的讨论，正如在第 9.4.2 小节中介绍过的，请参见文献 Hsu 和 Nelson(1990)、Hesterberg 和 Nelson(1998)。

11.5 间接估计法

这种 VRT 是为排队类型仿真开发的，用于被估计的量为稳态的性能度量，例如 d，w，Q 和 L 这种场合(参见附录 1B)。对于这一类模型，已经证明可获得方差减小[参见文献 Law(1974，1975)、Carson 和 Law(1980)、Glynn 和 Whitt(1989)]，但该思想也可用于其他类型的仿真中；还有，初步试验可以告知使用此方法是否能获得有意义的方差减小。其基础工具是在附录 1B 中给出的 d，w，Q 和 L 之间的理论关系。

令 D_i和 W_i分别是第 i 名到达 $GI/G/S$ 队列的顾客在队列中的延误时间和在系统中总的等待时间。因此，若 S_i是第 i 名顾客的服务时间，则 $W_i=D_i+S_i$。还有，令 $Q(t)$和 $L(t)$分别是系统在 t 时刻队列中的顾客数和系统中的顾客数。在一次仿真中，固定完成其服务的顾客数量为 n，并且持续 $T(n)$个单位的仿真时间，由此 d，w，Q 和 L 的直接估计分别为：

$$\hat{d}(n)=\frac{1}{n}\sum_{i=1}^{n}D_i,\quad \hat{w}(n)=\frac{1}{n}\sum_{i=1}^{n}W_i$$

$$\hat{Q}(n)=\frac{1}{T(n)}\int_0^{T(n)}Q(t)\,\mathrm{d}t,\quad \hat{L}(n)=\frac{1}{T(n)}\int_0^{T(n)}L(t)\,\mathrm{d}t$$

此时，$\hat{w}(n)=\hat{d}(n)+\overline{S}(n)$，其中，$\overline{S}(n)\sum_{i=1}^{n}S_i/n$ 且 $E[\overline{S}(n)]=E(S)$，即已知的期望服务时间。因此，w 的另一个估计值为：

$$\widetilde{w}(n)=\hat{d}(n)+E(S)$$

也就是说，我们将 $\overline{S}(n)$用它已知的(且零方差)期望值 $E(S)$来替代。我们称$\widetilde{w}(n)$为 w 的间接估计，且一个似乎合理的猜测是$\widetilde{w}(n)$会小于$\hat{w}(n)$，这是因为$\hat{w}(n)$中的随机项 $\overline{S}(n)$被确定的数值 $E(S)$代替来求得$\widetilde{w}(n)$。正如 Law(1974)证明过的那样，对于任意 $GI/G/S$ 队列和任意 n，情况确实如此，虽然这个证明并不像看起来那么简单，因为$\overline{S}(n)$和$\hat{d}(n)$并不是独立的。因此，间接估计$\widetilde{w}(n)$好于更显而易见的直接估计。

上一节中所建议的方差减小方法是通过附加关系 $w=d+E(S)$来实现的，而且显而易见，在 w 的估计器中，用随机变量 $\overline{S}(n)$是没有什么意义的，因为此时我们能使用其期望值 $E(S)$来替代，从而避免了额外的变化源。从下面两个恒等式的乘法关系获得 Q 和 L 的更好间接估计器恐怕就没有那么直观了：

$$Q=\lambda d \tag{11.4}$$

$$L=\lambda w \tag{11.5}$$

其中，λ 是到达速率(参见附录 1B)，它也许可以在仿真中得到。

式(11.4)给出的 Q 的间接估计器为：

$$\widetilde{Q}(n)=\lambda\,\hat{d}(n)$$

而且 Carson(1978)，Carson 和 Law(1980)证明了$\widetilde{Q}(n)$的方差渐近地[随着 n 和 $T(n)$都趋于无穷大]小于直接估计器$\hat{Q}(n)$。类似地，使用式(11.5)和此前的 w 优秀的间接估计器$\widetilde{w}(n)$，我们可以证明间接估计器：

$$\widetilde{L}(n)=\lambda\,\widetilde{w}(n)=\lambda[\hat{d}(n)+E(S)]$$

具有的方差渐进地小于直接估计器$\hat{L}(n)$。因此我们看到，用$\hat{d}(n)$的简单的确定性函数来估计 w，Q 和 L 好于直接估计它们。这就是在我们的贯穿本书的例子中我们强调在队列中延误时间的原因之一。间接估计的另一个优势是，在仿真过程中只需要搜集延误时间 D_1，D_2，…，而不需要搜集 W_i，$Q(t)$或 $L(t)$，即使我们确实要估计 w，Q 和 L。

例 11.16 对于 $M/G/1$ 队列，通过使用$\widetilde{Q}(n)$间接地估计 Q(例如)而不是使用$\hat{Q}(n)$直接地估计，准确的渐进方差减小可以计算出来[参见 Law(1974)]。对于指数分布、4-厄兰分布、超指数分布(参见第 4.5 节)服务时间，以及对于$\rho=0.5$，0.7，和 0.9，表 11.10 给出了它们的方差减小(百分数)。

表 11.10　Q 的间接估计的准确渐进方差减小，$M/G/1$ 队列

服务时间分布	减小/%		
	$\rho=0.5$	$\rho=0.7$	$\rho=0.9$
指数分布	15	11	4
4 厄兰分布	22	17	7
超指数分布	4	3	2

◀

上述的间接估计技术的一个弱点在于，当 $\rho\to1$ 时，方差的减小会减小到 0。看看表 11.10中的各行这是显而易见的，而且对于 $M/G/1$ 队列，Law(1974)解析地证明了这一点。但是由于高拥塞系统也是高可变的，在这种情况下我们需要方差减小是最大的。Carson(1978)开发了间接估计器的更一般的使用，这种用法与 Heidelberger(1980)改进的技术有关，在 ρ 接近 1 时做得较好，且一般获得较强的方程减小。再次用估计 Q 的例子，

我们注意到，有两个估计器$\hat{Q}(n)$和$\widetilde{Q}(n)$，可以将它们组合起来以得到一个线性组合估计器$Q(a_1, a_2, n)=a_1\hat{Q}(n)+a_2\widetilde{Q}(n)$，其中，$a_1+a_2=1$；$a_1$ 和 a_2 不必都是非负数。那么，在 $a_1+a_2=1$ 的条件约束下，可以选择 a_1 和 a_2 以最小化 $\text{var}[Q(a_1, a_2, n)]$。注意，$Q(1, 0, n)=\hat{Q}(n)$，且 $Q(0, 1, n)=\widetilde{Q}(n)$，从而该方法作为特别情形同时包括了直接估计和间接估计，这样，对于最优的(a_1, a_2)，有 $\text{var}[Q(a_1, a_2, n)]\leqslant\min\{\text{var}[\hat{Q}(n)], \text{var}[\widetilde{Q}(n)]\}$。但是，正如使用 CV 法那样(第 11.4 节)，最优的(a_1, a_2)需要估计。Carson (1978)基于再生法，给出了一个渐进有效的估计最优(a_1, a_2)的方法，并允许两个以上的不同估计器；参见第 11.4 节中有关使用多个估计器的讨论。Carson 的分析和实验研究表明，与直接估计器相比，往往达到至少 40%的方差减小。

其他讨论间接估计的文章有 Srikant 和 Whitt(1999)以及 Wang 和 Wolff(2003，2005)。

11.6 调节法

我们考虑的最后一个 VRT 是调节法，它与间接估计法有一个相同的特性，就是，我们利用模型的一些特性，用一个量的准确解析值来代替其估计值。在消除这个可变性的源中，我们希望，最终的输出随机变量将会更加稳定，虽然对这一点没有绝对的保证；还有，将调节法与直接仿真进行比较的初步试验方法可以揭示方差是否减小，以及减小到什么程度。我们将讨论的一般调节法的思想与 Hammersley 和 Handscomb(1964)给出的“条件蒙特卡罗”方法相同。

像通常做法一样，令 X 是一个输出随机变量，例如，一名顾客在队列中的延误时间，我们希望估计它的期望值 μ。假设有某个其他随机变量 Z，已知 Z 的任意特别的可能值 z，我们能解析地计算条件期望 $E(X|Z=z)$。注意，$E(X|Z=z)$是实数 z 的(已知的)确定型函数，但 $E(X|Z)$是随机变量，它与随机变量 Z 有相同的函数。那么，通过对 Z 的调节[参见，例如 Ross(2003，第 106 页)]，我们看到 $\mu=E(X)=E_z[E(X|Z)]$(外期望记做 E_z 是因为它具有对 Z 的分布)，所以随机变量 $E(X|Z)$对 μ 来说也是无偏的。举例来说，如果 Z 是离散的，具有概率密度函数 $p(z)=P(Z=z)$，则

$$E_Z[E(X|Z)] = \sum_z E(X|Z=z)p(z)$$

其中，我们假设 $p(z)$未知。此外，[参见，例如文献 Ross(2003，第 118 页)]

$$\text{var}_Z[E(X|Z)] = \text{var}(X) - E_Z[\text{var}(X|Z)] \leqslant \text{var}(X) \tag{11.6}$$

上式表明了如果我们观测随机变量 $E(X|Z)$(由 Z 中的一个观测值 z 计算得到)来代替直接观测的 X，我们将获得更小的方差。换句话说，做法是，我们仿真得到 Z 中的一个随机观测值 z(由于其分布是未知的)，并将该观测值放入已知公式 $E(X|Z=z)$中，并用此作为一个基本观测值；当然，在终止型仿真的情形下，这整个做法可以重复若干次。

自然，这里的技巧是规定随机变量 Z，使得：

- Z 可易于且有效地产生，因为我们必须继续对其进行仿真。
- $E(X|Z=z)$作为 z 的一个函数，应该对于 Z 能取值的任意可能值 z 都能解析并有效地计算。
- $E_z[\text{var}(X|Z)]$是大值，从而吸收式(11.6)中大部分的 $\text{var}(X)$。用口语说，$E_z[\text{var}(X|Z)]$是 X 对于 Z 的任意可能取值的平均条件方差，而且由于我们有 $E(X|Z=z)$ 的公式，我们对于给定的 Z 无须对 X 仿真，因此我们不受其方差影响。

由于这种 VRT 如此依赖于模型，我们的说明介绍了两个来自文献的成功实现。下面的例子来自 Lavenberg 和 Welch(1979)，说明使用调节法以获得排队网络中若干期望延误时间的估计中的方差减小。

例 11.17 某一分时计算机模型有单一 CPU 和 15 个终端，以及一个磁盘驱动器和一个磁带驱动器，如图 11.9 所示。在每一个终端处坐着一名用户，他的“思考”时间数量

是指数分布的，均值为 100 秒；之后向计算机发送一个作业，在那儿该作业加入 CPU 的 FIFO 队列。每一个进入 CPU 的作业占用 CPU 的时间数量为均值 1 秒的指数分布。作业离开 CPU 并完成的概率为 0.2，它返回其终端开始另一个思考时间；否则，该作业也许需要访问磁盘驱动器(以 0.72 的概率)或磁带驱动器(以 0.08 的概率)。去磁盘驱动器的作业隐含着在那儿的一个 FIFO 队列中等待，然后占用磁盘驱动器一个指数量时间 S_D(均值为 1.39 秒)，之后它必须返回到 CPU。类似地，从 CPU 出来去磁带驱动器的作业面对那里的一个 FIFO 队列，使用磁带驱动器一个均值为 12.50 秒的指数量时间 S_T，之后返回 CPU。所有的思考时间、服务时间和分支决策都是独立的，并且所有作业最初都处于在其终端处的思考状态。目标是要估计 d_C，d_D和 d_T，分别为作业在 CPU、磁盘驱动器和磁带驱动器的在队稳态期望延误时间。为此目标，运行长度取 400 个作业被处理并返回它们的终端所需的时间。

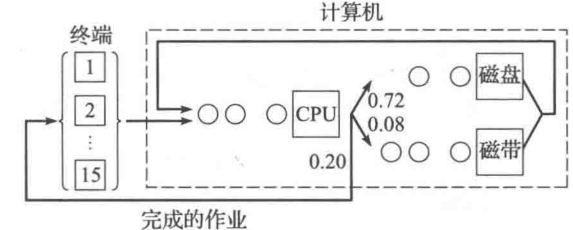


图 11.9　例 11.17 的分时计算机模型

可使用直接仿真来获得这些稳态期望延迟时间的估计，办法是简单地使用三个队列的每个观测到的延误时间的均值。但是，磁带驱动器延误时间的观测结果的数量会相当少，因为只有 8%的退出 CPU 的作业进入这一队列；因此，d_T的直接估计器，只基于相对少的数据，会有很大的可变性。

我们可以获得 d_T的一个不同估计，办法是，在每个作业离开 CPU 时，无论它去哪里，观测在磁带驱动器的任务(包括在队列中的和在服务中的)的总数 N_T。假定一个离开 CPU 的作业可能要去磁带驱动器，则它在队列中的期望延迟为 $E(S_T)N_T=12.50N_T$(因为，如果有正在磁带驱动器的服务中的作业，那么它具有指数分布的服务时间，正如习题 4.26 中说明的那样，由于指数分布的无记忆性的特点，它的剩余服务时间也是一个均值为 12.50 的指数分布。调节法对非指数分布的服务时间依然适用，但需要更多的信息)。用这种办法我们从每个离开 CPU 的作业得到一个对磁带驱动器延误时间的观测结果，而不是只从实际进入磁带驱动器队列的 8%的作业中得到。此外，我们使用了准确值 $E(S_T)=12.50$，而不是在直接仿真中产生的随机变量 S_T的抽签。对观测值 $12.50N_T$取平均会获得一个改进的 d_T的估计。由于离开 CPU 的作业，无论其是否实际进入磁带驱动器，都“看到”相同状态，因此该方法是有效的。

根据我们早前的一般讨论，我们进行仿真以观察一个作业一旦离开 CPU 时的 $Z=N_T$的值，以及 E(在磁带驱动器队列的延误时间$|N_T=z)=12.50z$，后者是在给定 $N_T=z$ 时的一个已知的确定型函数。然而，最终的输出变量，即磁带驱动器队列的平均延误时间，是整个动态仿真的更为复杂的最终结果，因此式(11.6)解决不了全部问题。虽然如此，我们还是希望该“局部”方差减小的一部分可以通过模型的动力学加以传播。

类似地，对于每个离开 CPU 的作业，我们能对该作业去磁盘驱动器进行调节，并取 $1.39N_D$作为对磁盘驱动器延误时间的观测值，其中，N_D为此时观察到在磁盘驱动器的在队列中和在服务中的作业的数量。这也会增加观测结果的数量，但不像磁带驱动器那样显著，因为无论如何几乎四分之三的退出 CPU 的作业要去磁盘驱动器。然而，我们使用已知的磁盘驱动器服务时间期望 S_D，而不是对它的随机观测。

最后，无论一个作业是离开其终端、磁盘驱动器，还是磁带驱动器，通过调节 N_C=在 CPU 的作业总数，我们可将 $1N_C$的值取平均(由于平均 CPU 时间为 1 秒)以力图得到 d_C的更好估计。此并不产生任何额外假想的 CPU 访问，因为无论如何所有这样的作业将进入 CPU，但它允许我们利用期望的 CPU 服务时间的知识。

在 100 次独立重复运行中，与直接仿真相比，对于 d_C，d_D和 d_T的估计值，分别观察

到19%，28%和56%的估计方差减小。正如所预期的，最大的受益是磁带驱动器队列，其中调节技术导致观察值的个数为直接仿真所观察的约12倍。该模型的其他七种版本也进行了仿真，还估计了二阶矩，依模型和估计对象的不同，调节法的方差减小为9%到86%之间。调节法中额外计算时间是可以忽略的。 ◀

正如上例所说明的，调节法VRT需要仔细分析模型的概率结构。而且，其成功最终不仅由于利用条件期望的解析公式的知识，而且还由于人为地增加了“稀有”事件的观测结果的数量，在上例中是作业去磁带驱动器。这一结论也适用于调节法下面例子的成功，是由Carter和Ignall(1975)给出的。

例11.18 开发了一个仿真模型以比较布朗克斯区派遣救火车的各种策略。某一类火灾分级为“严重”，因为有很大的危险造成生命损失且财产损失严重，除非消防部门能派去足够的消防车快速响应。仿真的目标是在给定的派遣策略下估计对严重火灾的期望响应时间。

历史数据表明，大约每30场火灾有1场是严重的。因此，模型必须完成大约30次仿真的火灾才能得到1个对严重火灾的响应时间的观测值，这样的话可能导致非常长且昂贵的运行来产生足够多的严重火灾，以得到对严重火灾的期望响应时间的好的估计。然而，此模型的特殊结构是：已知任一时刻的系统的状态(即所有救火车的位置)，对严重火灾的真的期望响应时间的期望可以解析计算，即它应该在那一时刻发生。此外，概率假设(严重火灾发生服从泊松过程)证明了在每一次观测系统状态时调节严重火灾事件是有效的，无论严重火灾是否实际发生过[参见Wolff(1982)]。因此，仿真将周期性的中断以观测系统状态，并根据当前的状态计算并记录对严重火灾的期望响应时间，并恢复仿真。最终的估计值是这些条件期望响应的均值，且包含的严重火灾数目比实际仿真的多了很多。

按照一般的讨论，仿真的目的是观测救火车位置的向量$\mathbf{Z}$，而E(对严重火灾的响应时间$|Z=z$)是解析得到的已知的z的函数，因此在仿真中不必估计。(假想的)严重火灾的发生频率超过实际观测到的，这是调节法的附加好处。

使用这种调节法，与直接仿真相比，估计的期望响应时间的方差减小了大约95%。调节-期望法的开销稍微大一些，不过即使考虑到这一点，在相同的运算量下方差减小92%。 ◀

注意，在动态仿真中，式(11.6)直接只能用于单个随机变量而不能用于仿真的总的输出随机变量(例如1 000名顾客在队列中的平均延误时间)，因此方差减小是不保证的。

还有一些其他的例子使用调节法作为也许有用的VRT。Carter和Ignall(1975)也考虑了库存模型，其中发生调节的事件是库存短缺，库存短缺很少发生，但一旦出现对系统的性能就有很大的影响。Burt和Garman(1971)考虑了随机PERT网络的仿真并对某一类任务时间进行调节，该时间对多于一条网络通路的情况是共有的；在他们使用小调节法中，人为地增加稀有事件的频率的概念没有出现。Garman(1972)、Sigal，Pritsker和Solberg(1979)讨论随机网络仿真中基于调节法的更深入的VRT。Minh(1989)提出了对于某些z，$E(X|Z=z)$是未知的情形下的广义调节法。

Nakayama(1994a，1994b)、Herdelberger(1995)、Shahabuddin(1994，1995)、Glasserman和liu(1996)、Glasserman(1999)、Nicola(2001)讨论了多种VRT，包括重要性采样。重要性采样可用于估计稀有但重要的事件的概率，例如一个高可靠性通信网络的瘫痪或排队系统中缓冲区溢出。

习题

11.1 一些VRT的几乎没有系统开销(例如CRN)，但其他的可能需要相当的额外的开销(例如外部CV法)，这样当决定一个特定的VRT是否值得时这是必须考虑的。令V_0为使用直接仿真的一个合适的方差度量(不使用VRT)并令V_1为相应的使用特定的VRT的方差度量。还有，令设C_0和C_1

分别为使用直接仿真和 VRT 方法进行特定长度的特定次数仿真所需的开销。请求 VRT 可用的关于 C_0、C_1、V_0 和 V_1 的条件。

11.2　在第 11.2.2 小节以及特别是图 11.1 中，我们考虑了对于给定的一对备选配置 CRN 是否会引入所要求的正相关性或者是否会造成相反的结果这样一个问题。考虑如下简单的蒙特卡罗例子，其中，U 表示一个随机数：

(1) $X_{1j}=U^2$ 和 $X_{2j}=U^3$；

(2) $X_{1j}=U^2$ 和 $X_{2j}=(1-U)^3$。

(a) 画出两个例子的响应图。

(b) 对于每一例子，解析地求 $\text{cov}(X_{1j}, X_{2j})$。

(c) 对于每一个例子，在独立采样和 CRN 两种情形下，解析地计算 $\text{var}(X_{1j}-X_{2j})$。

(d) 设计和执行一个小的仿真研究，实验性地验证你在(b)和(c)中的计算。

11.3　在第 11.3 节中我们讨论了 AV 法可能有效或事与愿违的条件。考虑如下简单的蒙特卡罗例子，其中 U 表示一个随机数：

(1) $X_j=U^2$；

(2) $X_j=4(U-0.5)^2$。

(a) 画出两个例子的响应图。

(b) 对于每一个例子，解析地求 $\text{cov}(X_j^{(1)}, X_j^{(2)})$。

(c) 对于每一个例子，在独立采样和 AV 法两种的情形下，解析地计算 $\text{var}[(X_j^{(1)}+X_j^{(2)})/2]$。

(d) 设计并执行一个小的仿真研究，实验性地验证你在(b)和(c)中得到的结论。

11.4　考虑图 11.10 所示的排队模型。顾客到达服从泊松过程，速度为 1 人/每分钟，且面对服务台 1 的先进先出队列，服务台 1 提供均值为 0.7 分钟的指数分布服务时间。退出服务台 1 后，顾客以概率 p 离去，并以概率 $1-p$ 去服务台 2。服务台 2 也按先进先出队列进入，并提供均值为 0.9 分钟的指数分布服务时间。所有服务时间、到达间隔时间和路径选择都是独立的。该系统最初为空且闲，它运行到 100 名顾客完成他们在队列中的总延误；进入服务台 2 的顾客在队列中的总延误时间是他或她在两个队列中延误时间之和。性能度量是完成在队列中的总延误的前 100 名顾客在队列中的期望平均总延误时间。

(a) 设此系统有两种配置，p 为 0.3 或 0.8。对每个系统进行 10 次重复运行，使用独立采样和 CRN 两种方法，并比较得到的性能度量差的估计值的估计方差。使用 CRN 时请注意保持恰当的同步。

(b) 对于 $p=0.3$，使用独立采样和对内 AV 法两种方法，进行 5 次成对运行，并比较估计的性能度量的估计方差。同样，请注意同步。

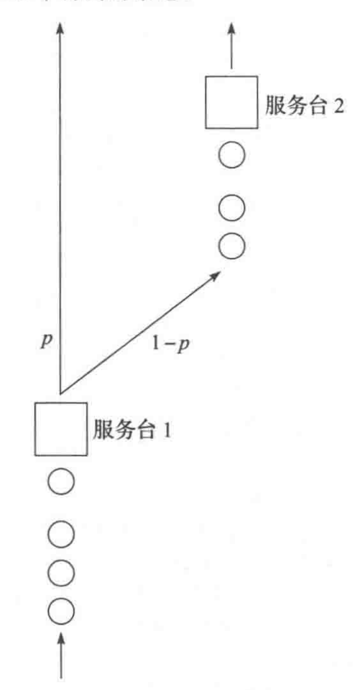


图 11.10　习题 11.4 的排队模型

11.5　回忆一下第 1.5 节的库存仿真，像在例 11.12 中为说明 AV 法那样来使用。有 3 个随机源(需求间隔时间、需求量和交货延迟)与 3 个成本成分(订货、维持与缺货)。请分析此模型，像在第 1.5 节中那样编程，以给出 AV 法的原理。特别地，看看如果一个小的(或大的)的随机数用作产生三类输入随机变量的每一类，该随机数对三类成本的每一类的影响是什么。例如，假设一个小 U 用于产生需求间隔时间，这一般会使订货成本变大还是变小，而其他则是一样吗?

11.6　回忆一下在第 2.6 节的银行模型，并假设银行管理者希望估计增加第六名和增加第七名出纳员好于图 2.36 所示结果的效果(与现有的 5 名出纳员配置相比)。请使用 CRN 做这件事，并对 3 个系统进行足够多次数的重复运行，以获得你感到在队列中的平均延误时间的期望值之差的估计足够精确。考虑产生顾客服务时间是在顾客到达时，而不是当他或她进入服务时。

11.7　回忆一下习题 2.19 的港口模型。

(a) 考虑对原来说明的模型应用 AV 法，特别是，哪些输入随机变量应该对偶地产生，以及如何

保持恰当的同步？

(b) 假设想要使用两台更快速的起重机来替换两台现有起重机。特别是，单台起重机对一条船的卸货时间是 0.2 天到 1 天之间的均匀分布的；其他条件仍然不变：讨论 CRN 的恰当应用与实现，以便将原来系统与建议的新系统进行比较。

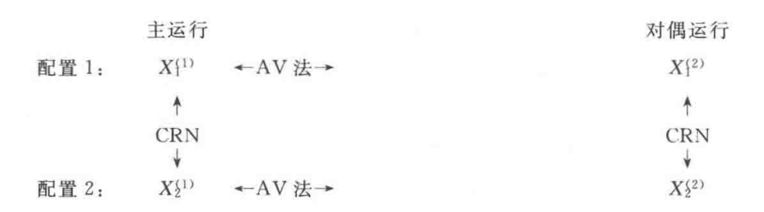
(c) 进行比较性仿真，同时使用独立采样和 CRN，并按需要进行重复运行以估计用 CRN 达到的方差减小(如果有的话)。

11.8 讨论使用 CRN 来比较习题 2.20 的计算机模型的作业处理策略(a)和(b)。

11.9 对于习题 2.22 的计算机模型的两种优先级策略，使用 CRN 锐化在这每策略下在队列中的期望平均延误时间的比较。

11.10 考虑如下两个 $M/G/1$ 队列，第一个队列具有指数服务时间，而第二个队列具有伽马服务时间，但两者具有的均值相同。讨论在实现 CRN 中随机数的同步问题，以比较两个队列，比如说，基于前 100 名顾客期望平均延误时间，已知初始条件为空且闲。第 8.2.1 小节和第 8.3.4 小节也许有用。

11.11 在比较两种系统配置中，考虑 AV 法和 CRN 一起用，即每一种配置的一对运行内使用 AV 法，而配置之间用 CRN。为简化问题，假设我们只进行每种配置的一对运行，示意性地表示如下：



$\zeta=E(X_1^{(l)}-X_2^{(l)})$的估计器是 $Z=(X_1^{(1)}+X_1^{(2)})/2-(X_2^{(1)}+X_2^{(2)})/2$。

(a) 根据相应于上面示意中的 4 次运行的方差和协方差求 $\text{var}(Z)$的表达式。

(b) 假设 CRN 和 AV 法自身都有效，即引入的相关性都有所要求的正负号，那么组合的方案有效吗？也就是说，与完全独立采样相比，我们知道该方案会减小 $\text{var}(Z)$吗？请解释。

11.12 若 Y_1，Y_2，…是独立同分布的随机变量序列，而 N 是一个与 Y_i以某种方式相关的正整数值的随机变量，那么，N 个 Y_i的样本均值不一定是 $\mu=E(Y_i)$的无偏估计。请使用这一事实来解释为什么在第 11.4 节的 CV 法的例子中，我们小心地将 CV 法的 Y 定义为固定数目的独立同分布随机变量的均值，例如到达间隔时间或服务时间，而不是令 Y 为在仿真结束前所产生的所有到达间隔时间或服务时间的均值。

11.13 对一般有 m 个控制变量的线性 CV 法来说，式(11.3)给出了控制估计器的方差，其中，权重 a_l必须指定。

(a) 对于 $m=2$ 和 3 的情形，求最优(使方差最小)权重。

(b) 假设控制变量彼此不相关，对于任意 m 求最优权重。

(c) 对于上述(a)、(b)两种情形，给出一种方法来估计最优权重。如果我们已知控制变量的方差(如果控制变量只是输入变量的均值，我们就会知道)，你的估计器能改进吗？在(a)问情形，如果我们还已知控制变量间的协方差，如何？

11.14 假设我们想要估计一个先入先出的 $M/G/1$ 队列前 100 名顾客在队列中的期望平均延误时间，其中初始条件为空且闲，平均到达间隔时间为 1 分钟，服务时间服从形状参数 $\alpha=2$ 和比例参数 $\beta=1.8/\sqrt{\pi}$的韦布尔分布。因此，平均服务时间是 $\beta\Gamma[(1/\alpha)+1]=(1.8/\sqrt{\pi})(\sqrt{\pi}/2)=0.9$ 分钟(参见第 6.2.2 小节)，且利用系数 $\rho=0.9$(参见第 8.3.5 小节中关于产生服从韦布尔变量的产生，用反变换法易于做到)。作为一个外部控制变量，我们可以使用 CRN 来对 100 个具有相同的平均到达间隔时间和服务时间的顾客来仿真 $M/M/1$ 队列，它精确地是例 11.13 的模型，并利用以下事实，该 $M/M/1$ 队列在队列中的期望平均延误时间已知，是 4.13。请利用第 11.4 节中给出的估计技术，由$n=10$次重复运行来来估计最优权重 a^*，并重复整个过程 100 次，以估计与直接仿真该 $M/G/1$ 队列相比的方差减小。方差减小值得吗？或者花时间对 $M/M/1$ 队列仿真代替对该 $M/G/1$队列进行附加的直接重复运行，这样做合算吗？

11.15 讨论对第 2.5 节的分时计算机模型恰当使用 AV 法的问题。对此模型的备选设计(例如买一块更快的 CPU，或改变服务定额)，CRN 能否适合用于进行比较性仿真？

11.16　在第 2.6 节的多出纳银行模型中讨论了允许换队的问题，这种换队是否会影响在队列中的顾客的平均延迟时间？为找到答案，将原始模型(有换队)视作“配置 1”，并定义“配置 2”为同样的模型但无任何换队；假设在每种情形下都有 5 名出纳。我们使用专用流来实现 CRN——流 1 用于到达间隔时间，流 2 用于服务时间。但是，我们应该如何由流 2 来产生服务时间这一点并不完全清楚；有以下两种(至少)可能：

(a) 当顾客到达时就产生其服务时间，并将服务时间作为一个属性与顾客一起存储起来。这物理上相应于这样一个思想，即强迫“相同”顾客(基于他们对出纳台的服务需求)在相同的时间到达两个配置。

(b) 只有当顾客开始服务时从流 2 产生服务时间，在这种情形下，两个系统将看不到相同的顾客，但两个配置的开始服务时间的所排顺序是相同的。

进行一个仿真试验，研究(a)或(b)哪种方式更好以实现 CRN。按两种配置下在队列中的期望平均延误时间差的估计器的方差进行比较。

11.17　假设我们希望对例 11.2 中的两个库存模型进行 5 次重复运行，如果我们将流 1、流 2 和流 3 用于重复运行 1；流 4、流 5 和流 6 用于重复运行 2 中，以此类推，5 次重复运行总共需要 15 个数据流。试问对 5 次重复运行仅用 7 个流我们如何能实现同步？使用 3 个流如何？

11.18　由于独立随机变量是不相关的(参见习题 4.8)，那为什么在表 11.3 中独立情况下(“I”行)X_{1j} 和 X_{2j} 之间的相关系数为 -0.17？

11.19　考虑例 9.25 的小型工厂，其中不合格零件的概率为 0.10。假设工厂正在考虑一个新的制造过程，不合格零件的概率将降低到 0.05。工厂希望基于在系统中的稳态平均时间来比较两个制造过程，并决定对每一过程进行 10 次独立重复运行，长度为 12 400 分钟，预热期为 2 400 分钟(40 小时)。令 X_{ij} 为过程 i($i=1$ 为原过程，$i=2$ 为建议的过程)第 j 次重复运行($j=1, 2, \cdots, 10$)的最后 10 000分钟在系统中的平均时间，并令 $Z_j=X_{1j}-X_{2j}$，对以下每种同步情形计算 Z_j 的样本方差：

(a) 两个制造过程完全独立仿真(无同步)。

(b) 只在新零件的到达间隔时间同步。

(c) 只在机器的故障和修理时间同步。

(d) 只在零件第一次通过系统过程中(如果需多次经过系统)的加工时间和检查时间同步。

以情形(a)作为基础情形，计算相应于情形(b)、(c)、(d)的方差减小情况，哪些随机变量对同步最重要。

11.20　在例 11.7 中，我们只使用一个随机数流，如何能对两种策略获得相同的需求间隔时间和需求量？

第12章 实验设计与优化

12.1 引言

本章介绍统计实验设计和优化的用途，其中“实验”是指一个计算机仿真模型执行。与第10章一样，我们将会讨论不同系统配置下的仿真，并检查与比较其结果。在第10章中，一方面我们假设不同的系统配置是直接给出的，可能是基于物理约束、合同义务或者政治方面的考虑，这些配置被指定为备选方案。而另一方面，在本章，在仿真研究目标中我们对结构考虑少一些；我们可能最想要找出何种参数和结构假设对系统性能影响最大或者何种模型参数会使系统达到最佳性能。为了达到这些更广义(或者说更远大)的目标，我们不能够像第10章中那样进行正式的统计分析或者在分析之后做出很精确的概率断言。

在实验设计术语中，组成一个模型的输入参数和结构假设称为因子，输出性能度量称为响应。至于将哪些参数和结构假设认为是模型中固定的部分以及哪些是实验因子的决策取决于研究的目的，而不是模型的内在形式。另外，在仿真研究中，通常有多个不同的响应或者目标性能度量。

因子可能是定量的，也可能是定性的(有时也称为分类的)。定量因子自然假设取数值，而定性因子描述性质上非量化的结构假设，某些例子如表12.1所示。

表12.1 因子和响应的例子

系统	可能的因子	定性或是定量	可能的响应
库存系统	重新订货点	定量	每月的平均成本，库存物品的平均件数
	最佳订购上限水平	定量	
制造系统	机器数目	定量	平均滞留时间，生产量、平均队长
	叉车数目	定量	
	叉车分配规则	定性	
通信网络	节点数	定量	平均端到端延迟，流量
	链路数	定量	
	路由规则	定性	

我们也可以将仿真实验中的因子分类为可控的和不可控的，这取决于它们表示的动作选择是否是相应的现实世界系统的管理者。通常我们在仿真实验中将关注可控因子，因为它们与实际系统实现所做的决策是最相关的。然而，在仿真实验中也可能对不可控因子感兴趣，例如，我们可能想看看顾客到达速率增加百分之十对系统拥塞的影响。在像仿真这样的数学建模活动中，无论现实世界的可控性如何，我们毕竟要控制所有的因子。

仿真中实验设计的主要目标是确定哪个因子对响应的影响最大，且为此用最少次数的仿真。这通常称为“析因实验”或者“灵敏度分析”(参见5.4.4节)。精心设计的实验效率要高得多，远超过胡乱运行、简单而不系统地尝试许多不同配置以观察发生了什么的做法。12.2节和12.3节分别讨论 2^k 析因实验和 2^{k-p} 分部析因设计，在实验的初期我们对哪些因子重要以及它们如何影响响应都一无所知时，这些方法都特别有用。

在我们得知哪些因子十分重要，并且知道它们如何影响响应后，我们可以基于重要的因子开发出元模型或者响应面，以完成下面的这些事情：

- 预测没有经仿真的系统配置的模型响应，因为这些仿真模型的执行时间很长。
- 使用响应面方法，搜索出优化(根据要求最大或者最小)某个响应的输入因子组合。

第 12.4 节讨论的元模型通常是一个回归方程的形式，它将模型的响应对应于一个或多个输入。

实验设计是统计领域的一个重要论题，本章不对其作完整的讨论。实际上，有很多书专门研究实验设计和响应面方法，包括 Barton(1999)；Box 和 Draper(2008)；Box、Hunter 和 Hunter(2005)；Montgomery(2013)；Myers 等(2009)；Hamada 和 Wu(2009)。在仿真的背景下讨论仿真实验的参考文献有 Barton(2010)，Cheng 和 Kleijnen(1999)、Cioppa 和 Lucas(2007)、Kleijnen(1988，2007，2008)、Kleijnen 等(2005)、Sanchez(2005)，以及 Sanchez 和 Sanchez(2012)。

第 12.5 节讨论用于优化仿真系统性能的更通用的方法。我们的重点将会是集成到商用仿真软件的优化软件包中的方法(如元启发式算法)。

尽管人们一般可能认为仿真实验只是实验的一个实例，但不同于传统上用于实验设计文献中的的通常的物理制造、实验室或者农业实验，仿真具有某些优点：

- 正如前面提到的，我们有机会控制像顾客到达速率这种现实不可控的因素。因此，相对于我们对系统做物理实验，我们可以研究的事件种类更多。
- 对仿真实验增强控制的另一方面的原因是随机数发生器的确定性(参见第 7 章)。那么，在仿真实验中，我们可以控制基本的变化源，而不是像物理实验中那样。因此，我们能通过公共随机数(CRN)用方差缩减技术(见第 11.2 节)，以使我们的结论更加鲜明，虽然必须小心避免潜在的意外(参见下面例 12.3 以及习题 12.3 的讨论)。
- 在大多数物理实验中，对随机化处理(因子组合)以及运行的顺序(应用随机化处理的次序)要小心，以避免实验条件带来的系统性偏差，例如在无绝热的生物实验过程中，实验室环境温度稳定上升。在仿真实验中，只要随机数发生器工作正常，就没有必要随机化。
- 对于某些物理实验，出于时间与费用的考虑，只能对因子水平的每个组合做一次重复运行。因此，为了判断一个特别的因子是否对响应在统计上有显著的影响，有必要假设因子水平的每个组合的响应都具有相同的方差，恐怕这是有问题的。然而，对很多(即使不是大多数)仿真模型来说，现在完全可能(由于计算机的速度)为每个输入因子组合做多次重复运行，结果是判断统计显著性的方法变简单了。

12.2　2^k 析因设计

如果一个模型只有一个因子，实验设计在概念上是简单的：我们只要在因子的各种值或者“水平”上运行仿真，还可能在每个因子水平上构造期望响应的置信区间。对于定量因子，响应作为因子水平的函数图可能比较有用。在终止型仿真的情况下(参见 9.4 节)，我们可以在每个因子水平进行一定数量 n 次的独立重复运行。通常至少会有两个因子水平，因此我们会需要 $2n$ 次重复运行。

现在假设有个 $k(k\geqslant 2)$ 因子，且我们想要得到每个因子如何影响响应的初始估计，我们可能还想判断因子之间是否彼此相互作用，即某个因子对响应的影响是否取决于其他因子的水平。测量某一特定因子的影响的一个办法恐怕就是将其余 $k-1$ 个因子固定在某一组值，并在感兴趣因子的每两个水平上运行仿真，观察响应对这个单一因子的变化是如何反应的，然后重复整个过程以检查其他的因子。这种策略称为单因子轮换(one-factor-at-a-time，OFAT)法，按照为了得到规定的精度需要运行仿真的次数来说，这种方法的效率相当低[参见文献 Montgomery(2013，第 4-5 页，第 186-187 页)]。更重要的是，它不允许我们检测任何因子之间的相互作用；的确，它假设不存在相互作用，这往往与仿真应用的情况不符。

例 12.1 假设我们有两个因子 A 和 B。假设这些因子的基准水平分别为 A^- 和 B^-。另外，令 A^+ 和 B^+ 为这些因子的建议水平。那么 OFAT 法就会规定按 AB 的以下三种组合仿真：

$$A^-, B^-$$
$$A^+, B^-$$
$$A^-, B^+$$

得到响应分别为 $R(A^-, B^-)$、$R(A^+, B^-)$ 与 $R(A^-, B^+)$。那么就会计算出因子 A 从 A^- 变化到 A^+ 对响应的效用为：

$$R(A^+, B^-) - R(A^-, B^-) \tag{12.1}$$

然而，这个计算是基于 B 因子的值为 B^- 水平。可是，如果 B 因子的值为 B^+ 水平的话，对改变因子 A 的响应的影响就可能会相当不同(即如果因子相互作用的话)，数值相关的例子参见例 12.3(类似讨论适用于因子 B 时)。

如果我们也仿真了 A^+、B^+ 的组合，并得到响应 $R(A^+, B^+)$，那么也能计算对改变因子 A 的响应的效用为：

$$R(A^+, B^+) - R(A^-, B^+) \tag{12.2}$$

然而，最后这个计算在 OFAT 法中就不可能，因为 A^+，B^+ 不会被仿真。对于下面要讨论的 2^k 析因设计，式(12.1)和式(12.2)给出的平均差将会用来估计因子 A 从 A^- 水平变化到 A^+ 水平的响应的效用。◀

有一个方便得多的方法来判断因子对响应的影响，还可以测量交互效用，称为 2^k 析因设计，该方法需要我们只从每个因子中挑选出两个水平，然后在 2^k 个可能的组合上，有时也称为设计点，调用仿真运行。通常我们将一个因子的水平记为负号，另一个记为正号；将哪个符号赋给哪个水平是任意的，但是对于定量因子，如果我们给数值较小的水平赋予负号，则会减少混淆。对水平的选择应当咨询领域专家，相互之间的距离应当足够大，使得我们可以看到响应之间的差别，但是也不要分开到产生没有意义的系统配置。因为对于每个因子都只使用了两个水平，假设在整个因子的范围内响应是近似线性的(至少是单调的。如果在该范围内响应是非单调的(例如抛物线形状)，那么我们可能会被误导认为因子对响应没有影响)。在第 12.4 节将会讨论一种测试线性假设的方法。

2^k 析因设计可以使用表格的形式简洁地表示出来，对于 $k=3$ 的情况如表 12.2 所示。变量 R_i，$i=1$，2，…，8 指的是对第 i 个因子水平的组合运行仿真的响应。例如，R_6 是因子 1 和因子 3 分别在它们的"+"水平上，而因子 2 在它的"−"水平上的运行仿真所得到的响应。我们一会就可以看到写下的这个矩阵，称为设计矩阵，方便了因子效用以及交互的计算。

表 12.2　2^3 析因设计的设计矩阵

因子组合(设计点)	因子 1	因子 2	因子 3	响应
1	−	−	−	R_1
2	+	−	−	R_2
3	−	+	−	R_3
4	+	+	−	R_4
5	−	−	+	R_5
6	+	−	+	R_6
7	−	+	+	R_7
8	+	+	+	R_8

因子 j 的主效用记为 e_j，在保持其他因子水平不变下，将因子 j 从其"−"水平变化到"+"水平时响应的平均变化。这个均值包括了设计中其他所有因子水平的组合。必须认识到主效用的计算只是相对于当前的设计和因子水平，除非满足其他条件(如没有交互效用)，我们的计算一般无法超出这点。关于主效用的解释的这些局限性在本节的后面讨论。

对于表 12.2 中的 2^3 析因设计，因子 1 的主效用为：

$$e_1 = \frac{(R_2 - R_1) + (R_4 - R_3) + (R_6 - R_5) + (R_8 - R_7)}{4}$$

注意到在设计点 1 和点 2，因子 2 和因子 3 固定不变，就像在设计点 3 和点 4、点 5 和点 6、点 7 和点 8 中一样。因子 2 的主效用为：

$$e_2 = \frac{(R_3 - R_1) + (R_4 - R_2) + (R_7 - R_5) + (R_8 - R_6)}{4}$$

而因子 3 的主效用为：

$$e_3 = \frac{(R_5 - R_1) + (R_6 - R_2) + (R_7 - R_3) + (R_8 - R_4)}{4}$$

观察表 12.2 以及上面关于 e_j 的表达式，可得到定义主效用的另外一种方法，也是一种更加简便的方法。从名字上看，e_j 就是因子 j 在其"+"水平的时候的平均响应与该因子在其"−"水平的时候系统平均响应的差值。因此，为了计算 e_j，我们可以简单地将"因子 j"栏的符号赋给对应的 R_j，将它们加起来，并除以 2^{k-1}(换句话说，如果我们将设计矩阵中的"+"号和"−"号分别理解为"+1"和"−1"，那么可以将"因子 j"列同"响应"列作点积，然后除以 2^{k-1})。例如，在表 12.2 中的 2^3 析因设计中，有

$$e_2 = \frac{-R_1 - R_2 + R_3 + R_4 - R_5 - R_6 + R_7 + R_8}{4}$$

同前面 e_2 的表达式完全相同。

主效用度量由于单个因子变化而引起的响应的平均变化，这个平均是对其他所有 $k-1$ 个因子的所有可能组合取的平均(共 2^{k-1} 个)。然而，因子 j_1 的影响有可能以某种方式依赖于某个其他的因子如 j_2 的水平，在这种情况下，这两个因子称为交互效用。交互效用的度量就是因子 j_2 在"+"水平(除 j_1、j_2 外的其他因子保持不变)时 j_1 的平均效用与因子 j_2 在"−"水平(除 j_1、j_2 外的其他因子保持不变)时 j_1 的平均效用之间的差值。习惯上将这个差值的一半称作"两因子(双向)交互效用"，记作 $e_{j_1 j_2}$，也称作"$j_1 \times j_2$ 交互效用"。例如，在表 12.2 的设计中，我们有：

$$e_{12} = \frac{1}{2}\left[\frac{(R_4 - R_3) + (R_8 - R_7)}{2} - \frac{(R_2 - R_1) + (R_6 - R_5)}{2}\right]$$

$$e_{13} = \frac{1}{2}\left[\frac{(R_6 - R_5) + (R_8 - R_7)}{2} - \frac{(R_2 - R_1) + (R_4 - R_3)}{2}\right]$$

并且

$$e_{23} = \frac{1}{2}\left[\frac{(R_7 - R_5) + (R_8 - R_6)}{2} - \frac{(R_3 - R_1) + (R_4 - R_2)}{2}\right]$$

例如，为了看看 e_{13} 的表达式如何度量了上面的文字描述的量，请注意表 12.2 的设计矩阵，对于设计点 5、点 6、点 7 和点 8，因子 3 一直都是处于"+"水平；而在设计点 5 与点 6 之间(其他因子，本例中只有因子 2，保持在"−"水平)，以及设计点 7 与点 8 之间(因子 2 保持"+"水平)，因子 1 从"−"水平跳到了"+"水平。因此，上面 e_{13} 表达式的方括号中的第一部分是因子 1 从"−"水平变化到"+"水平，此时因子 3 保持"+"水平的平均效用。类似的，方括号中的第二部分是因子 1 从"−"水平跳至"+"水平(设计点 1 到点 2，点 3 到点 4)，此时因子 3 保持"−"水平的平均效用。那么，这两个部分的差值就是因子 1 依赖于因子 3 是"+"水平还是"−"水平的响应的效用的差值；这个差值的一半就是因子 1 和因子 3 之间的交互效用的定义。

当采用主效用时，基于设计矩阵有个更简单的方法计算交互效用。如果我们将 e_{13} 的表达式重新排列，例如，R_i 按照 i 的增序排列，我们得到：

$$e_{13} = \frac{R_1 - R_2 + R_3 - R_4 - R_5 + R_6 - R_7 + R_8}{4}$$

现在如果我们新增一列，记为“1×3”，包含 8 个符号，将“因子 1”列的第 i 个符号与“因子 3”列的第 i 个符号相乘(同号相乘为“+”，反号相乘为“−”)，我们就可以得到一列符号，正好是形成 e_{13} 所用的 R_i 符号，与主效用的做法一样，除数为 2^{k-1}。因此，因子 1 和因子 3 的交互效用可以认为是因子 1 和因子 3 处于相同水平时(同为“+”或同为“−”)的平均响应，以及它们取相反水平时的平均响应之间的差值(我们把以这种方法计算 e_{12} 和 e_{23} 的事情留给读者)。注意，两因子交互效用是完全对称的；例如，$e_{12}=e_{21}$，$e_{23}=e_{32}$，等等。

我们可以定义并计算 3 个以及更多因子的交互效用，一直到 k 因子交互效用，尽管解释越来越困难。例如，在表 12.2 中的 2^3 析因设计，三因子交互效用为因子 3 取“+”水平时因子 1、因子 2 的交互效用和因子 3 取“−”水平时因子 1、因子 2 的交互效用之间的差值，也就是：

$$e_{123}=\frac{1}{2}\left[\frac{(R_8-R_7)-(R_6-R_5)}{2}-\frac{(R_4-R_3)-(R_2-R_1)}{2}\right]$$
$$=\frac{-R_1+R_2+R_3-R_4+R_5-R_6-R_7+R_8}{4}$$

e_{123} 的第二个表达式是通过符号乘积得到的，将表 12.2 中因子 1、因子 2 和因子 3 列的第 i 个符号相乘，并赋给 R_i；分母还是 2^{k-1}。三个及三个以上因子的交互效用也是对称的，$e_{123}=e_{132}=e_{213}$，等等。

如果两个或两个以上交互效用已经得到，那么在一个有意义的交互效用中所包括的每个因子的主效用一般不能简单地解释为将因子从“−”水平转移到“+”水平时的效用，因为响应变化的大小以及可能符号至少取决于另外一个因子的水平。

例 12.2 稍微改变订货策略，对第 1.5 节库存模型重新参数化是很方便的。特别是，我们像以前那样令 s 为重新订货点，但不用订货至上限 S，我们根据差值 $d=S-s$ 来观察决策。换句话说，我们的实验因子为 s 和 d，并且我们感兴趣的是它们如何影响期望平均总体运营成本；当然，S 就是 $s+d$。我们选择这些因子的“低”水平和“高”水平如表 12.3的编码所示[如果我们在编码图中使用了原来的参数 s 和 S，那么就会得到如(20，10)这样的无意义库存策略]。表 12.4 给出了设计矩阵、对应的响应值，以及一个附加列，用于表示计算 $s\times d$ 交互效用的符号。每个 R_i 都是单一 120 个月重复运行的月平均成本；对于每个 R_i，我们都使用了独立的随机数流。主效用为：

表 12.3 库存模型 s 和 d 的编码

因子	−	+
s	20	60
d	10	50

$$e_s=\frac{-144.16+144.50-119.99+147.00}{2}=13.68$$

$$e_d=\frac{-144.16-144.50+119.99+147.00}{2}=-10.84$$

表 12.4 库存模型关于 s 和 d 的 2^2 析因设计的设计矩阵和仿真结果

因子组合(设计点)	s	d	s×d	响应
1	−	−	+	144.16
2	+	−	−	144.50
3	−	+	−	119.99
4	+	+	+	147.00

而 $s\times d$ 交互效用为：

$$e_{sd}=\frac{144.16-144.50-119.99+147.00}{2}=13.34$$

因此，s 从 20 升到 60 的平均效用就是增加了 13.68 的月成本，而 d 从 10 升到 50 的平均效用就是月成本平均减少了 10.84。因此，小 s 值和大 d 值可能会比较好，因为希望月成

本低。因为 $s \times d$ 的交互效用为正，因此这进一步说明将 s 和 d 取相反的水平可能会得到较低的成本。然而，如果这个交互效用的确非常明显(例 12.3 中提到了这个问题)，那么 s 对响应的影响取决于 d 的水平，反过来也是一样的。◀

由于 R_i 是随机变量，效用也是随机的。为了找出效用是否是“实际的”随机，而不是用采样波动来解释的，我们必须判断效用是否为统计显著的。这是实验设计的文献中常常提到的，通过方差分析的方法可以完成[例如，参见 Montgomery(2013，第 245-249 页)]。方差分析假设响应对于每个设计点，都具有相同的总体方差。然而，正如我们将在例 12.3和例 12.4 中看到的那样，在仿真建模中这常常不是一个好的假设。因此，我们将采用简单的方法，重复运行整个设计 n 次，以得到每个效用的 n 个独立值。这些数值可以用来构建满足期望效用的置信区间。例如，对于 $i=1, 2, \cdots, n$，令 e_j^i 是重复运行 i 次时因子 j 的主效用，令：

$$\overline{e}_j(n) = \frac{\sum_{i=1}^{n} e_j^i}{n}$$

且

$$S_j^2(n) = \frac{\sum_{i=1}^{n} [e_j^i - \overline{e}_j(n)]^2}{n-1}$$

然后可利用这些数据依照式(4.12)采用 $n-1$ 自由度的 t 分布为期望效用构建 $100(1-\alpha)\%$ 的置信区间。

$$\overline{e}_j(n) \pm t_{n-1,1-\alpha/2}\sqrt{S_j^2(n)/n}$$

如果一个特定效用的置信区间不包含 0，那么我们得出结论，这个效用是统计显著的；否则，我们没有统计的证据讲这是实际存在随机的。一般情况下，n 越大，使得置信区间宽度越小，越容易判断出一个效用是否是统计显著的。我们也必须记住，效用的统计显著性并不必然意味着效用大小实际上是显著的。

例 12.3 我们对例 12.2 中库存模型的整个 2^2 析因设计重复运行 $n=10$ 次，表 12.5 给出了每 4 个设计点整个 10 次重复运行的响应的样本均值和方差。注意，最大的和最小的样本方差的差别一个因子接近 14。基于我们得到的三个效用的每一个 10 次独立重复运行，表 12.6 给出了 $E(e_s)$、$E(e_d)$、$E(e_{sd})$ 的 96.667%置信区间。因此，根据邦费罗尼(Bonferroni)不等式(参见第 9.7 节)总置信区间水平至少为 90%。所有的影响效用看起来都是真实的，因为它们的置信区间都不包含 0。

表 12.5　库存模型响应的采样均值和方差

设计点	采样均值	采样方差
$s=20$，$d=10$	135.71	22.24
$s=60$，$d=10$	143.94	2.26
$s=20$，$d=50$	119.45	15.07
$s=60$，$d=50$	148.17	1.60

表 12.6　期望效用的 96.667%置信区间，库存模型

期望效用	96.667%置信区间
$E(e_s)$	18.47 ± 2.58
$E(e_d)$	-6.02 ± 2.47
$E(e_{sd})$	10.25 ± 2.88

在图 12.1a 和图 12.1b 中，我们给出了 s 和 d 的主效用图。对于每个图，在感兴趣的因子的某个特定水平(“—”或“+”)的平均成本是表 12.5 中其他因子两个水平上的样本均值的平均值(因此，图 12.1a 中的 127.8 是 135.71 和 119.45 的均值)。如果我们能用文

字解释这些主效用，我们会期望当我们将 s 从 20 变到 60 时月平均成本将会增加 18.47，而当我们将 d 从 10 变到 50，月平均成本将会降低 6.02。然而，由于 s 和 d 之间有显著的交互效用，这些主效用实际上是有限值。

图 12.2 给出了 s 和 d 的交互效用图，其中非平行线表明了存在显著的交互效用。特别是，当 $d=10$ 时，s 从 20 变化到 60 会增加 8.23 的平均成本(参见表 12.5)。然而，当 $d=50$时，s 从 20 变化到 60 会增加 28.72 的平均成本。注意，28.72 和 8.23 差值的一半是 10.25，也就是 $\overline{e}_{sd}$(10)(10 个交互效用的均值)。注意，当 S 从 20 移动到 60 时，由 OFAT 方法得到的平均成本增加了 8.23，而从 2^2 因子设计中得到的是 18.47。

我们从图 12.2 得出结论，s 和 d 对平均月成本都具有显著的效用。然而，s 变化引起平均成本的实际数值变化依赖于 d 的水平，反过来也是一样，这将在第 12.4 节进一步讨论。

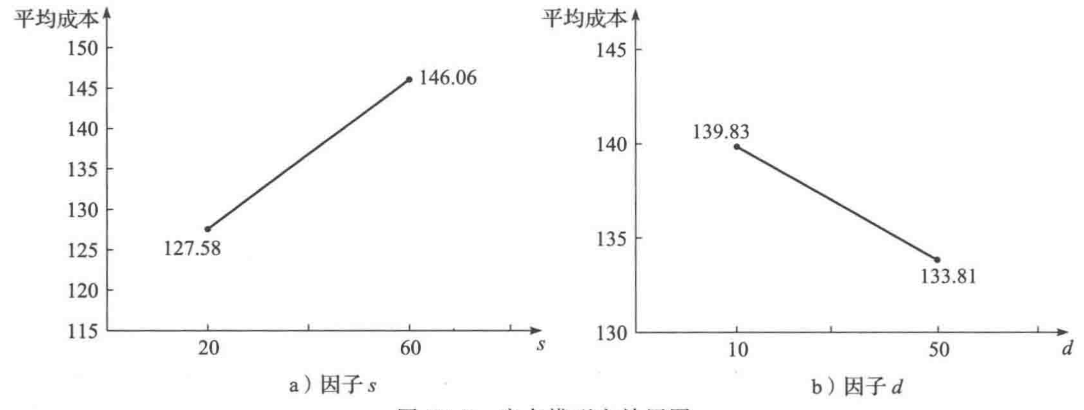


图 12.1 库存模型主效用图

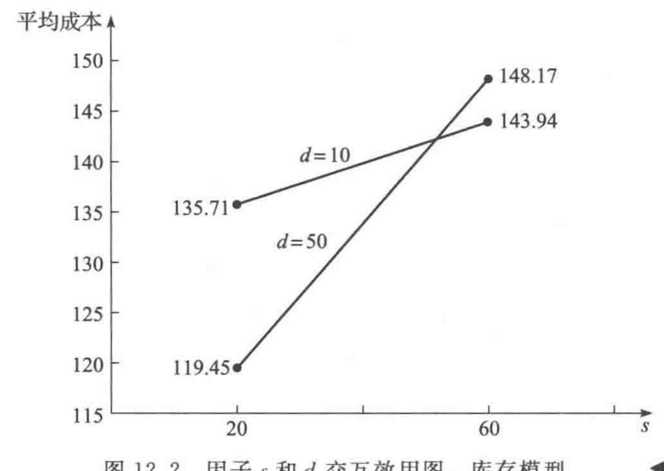


图 12.2 因子 s 和 d 交互效用图，库存模型 ◀

在例 12.2 和例 12.3 中我们对四个不同的因子水平组合独立地进行了仿真实验。由于这里处理的是四个不同的配置，所以可以对这四个使用公共随机数(CRN，见第 11.2 节)，以力图减小期望效用的置信区间的半长。然而，这里的情况不像在 11.2 节中那么简单。如果 CRN 实际上有效，并减少了不同配置的响应之间所希望的正关联，某些协方差就会带错误的符号进入效用方差的表达式中，方差就会因此而增加或减小，取决于协方差的相对大小，以及包括的效用是哪一个(参见习题 12.3)。我们使用 CRN 方法重新运行过例 12.3 中的实验，且发现 $E(e_s)$、$E(e_d)$、$E(e_{sd})$的置信区间半长分别减小了 1.31、0.68 和 0.46。这个现象是仿真中的随机数分配的整个问题的一个例子，最初是由 Schruben 和 Margolin(1987)提出的，也可参见 Hussey，Myers 以及 Houck(1987)。

本节最后一个例子是关于一个具有六个因子的模型，它展示了对因子数目多的问题应用 2^k 析因设计所需的计算复杂度。

例 12.4 例 9.25 介绍了一个小型工厂模型，模型中零件到达加工，然后去检查；检查不合格的零件返回加工站重新加工(参见图 9.9)。机器还有可能失效停机且必须进行维修。模型运行 160 个小时，前 40 小时的数据删除(参见例 9.33)，使得该模型预热到稳定状态。我们将零件花费在系统中的平均时间作为我们的响应。由于对于任何好的(即长时间稳定运行)系统配置，每小时也只生产 60 个零件，因此不考虑生产量。零件到达服从泊松过程，速率为每分钟 1 个，我们假设这个因子不可控制，在我们的设计中不包括这个实验因子。有六个别的可控因子，如表 12.7 所示。对于每个因子，如例 9.25 中描述的一样，“—”水平为当前的状况，“+”水平表示在减少零件在系统中的平均时间方面可能有改进，但以这种方式编码每个主效用将会是对应于声称的改进的结果。因子 1 到因子 5 是定量因子，每种情况的“+”水平产生 10%的改进；因子 6 是定性因子，表示排队规则的变化，两个队列的每一个按照零件的实际加工时间或检查时间，从 FIFO 变到最短加工时间的作业优先。

表 12.7 因子编码，小型工厂模型(所有的时间以分钟为单位)

因子编号	因子描述	—(当前)	+(增加后)
1	加工时间	$U(0.65, 0.70)$	$U(0.585, 0.630)$
2	检查时间	$U(0.70, 0.80)$	$U(0.675, 0.720)$
3	机器正常运行时间	expo(360)	expo(396)
4	机器维修时间	$U(8, 12)$	$U(7.2, 10.8)$
5	坏零件的概率	0.10	0.09
6	队列规则(两个)	FIFO	最短加工时间的作业优先

◀

表 12.8 所示的是 2^6 析因设计的整个设计矩阵，需要 64 个不同的设计点。重复运行整个设计 $n=5$ 次，以得到期望效用的置信区间，如例 12.3 所示，因此，该实验共 320 次仿真运行。自然，整个 64 个设计点使用 CRN 是最方便的。

表 12.8 2^6 析因设计的设计矩阵，小型工厂模型

设计点	因子编号					
	1	2	3	4	5	6
1	—	—	—	—	—	—
2	+	—	—	—	—	—
3	—	+	—	—	—	—
4	+	+	—	—	—	—
5	—	—	+	—	—	—
6	+	—	+	—	—	—
7	—	+	+	—	—	—
8	+	+	+	—	—	—
9	—	—	—	+	—	—
10	+	—	—	+	—	—
11	—	+	—	+	—	—
12	+	+	—	+	—	—
13	—	—	+	+	—	—
14	+	—	+	+	—	—
15	—	+	+	+	—	—
16	+	+	+	+	—	—
17	—	—	—	—	+	—
18	+	—	—	—	+	—
19	—	+	—	—	+	—

（续）

设计点	因子编号					
	1	2	3	4	5	6
20	+	+	−	−	+	−
21	−	−	+	−	+	−
22	+	−	+	−	+	−
23	−	+	+	−	+	−
24	+	+	+	−	+	−
25	−	−	−	+	+	−
26	+	−	−	+	+	−
27	−	+	−	+	+	−
28	+	+	−	+	+	−
29	−	−	+	+	+	−
30	+	−	+	+	+	−
31	−	+	+	+	+	−
32	+	+	+	+	+	−
33	−	−	−	−	−	+
34	+	−	−	−	−	+
35	−	+	−	−	−	+
36	+	+	−	−	−	+
37	−	−	+	−	−	+
38	+	−	+	−	−	+
39	−	+	+	−	−	+
40	+	+	+	−	−	+
41	−	−	−	+	−	+
42	+	−	−	+	−	+
43	−	+	−	+	−	+
44	+	+	−	+	−	+
45	−	−	+	+	−	+
46	+	−	+	+	−	+
47	−	+	+	+	−	+
48	+	+	+	+	−	+
49	−	−	−	−	+	+
50	+	−	−	−	+	+
51	−	+	−	−	+	+
52	+	+	−	−	+	+
53	−	−	+	−	+	+
54	+	−	+	−	+	+
55	−	+	+	−	+	+
56	+	+	+	−	+	+
57	−	−	−	+	+	+
58	+	−	−	+	+	+
59	−	+	−	+	+	+
60	+	+	−	+	+	+
61	−	−	+	+	+	+
62	+	−	+	+	+	+
63	−	+	+	+	+	+
64	+	+	+	+	+	+

图 12.3 将每个重复运行的响应(以分钟为单位)绘制为小圆点，因此，对于每个设计点，有五个垂直分布的圆点。大的圆点表示每个设计点的五次重复运行的均值，而水平线给出了响应的总平均值，即所有 320 次单个重复运行结果的均值。从图中可以直接观察出如下几个问题：

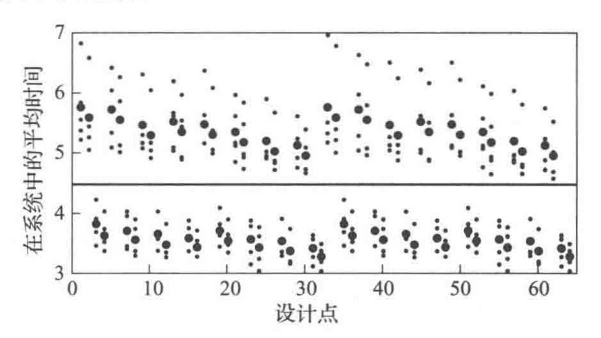


图 12.3 小型工厂实验设计：单个重复运行以及重复运行结果的平均值

- 大圆点强一致地成对——两个“大”值，接着是两个“小”值，接着又是两个“大”值，等等。回过头看看表 12.8 所示的设计矩阵，我们看到这个模式按因子 2 也就是检查时间因子的水平变化而变化。因此看来很明显(下面通过效用计算将会正式地证实)减小检查时间会对响应产生一致并且可观的改进。
- 在每两个大圆点组中，第二个低一些。因子 1(加工时间)在每个点均改变水平，因此我们看到通过减小加工时间能够减小在系统中的平均时间。
- 在每 16 个大圆点块中，后 8 个比前 8 个低。因子 4(机器维修时间)每 8 个点改变水平。因此我们看到减少机器故障停机时间可以期望改进性能。
- 在每 32 个大圆点块中，后 16 个点更低，表明减小零件未通过检查的概率的好处(因子 5，每过 16 设计点它切换一次水平)。
- 设计点 33 到点 64 的结果看起来几乎是设计点 1 到点 32 的硬拷贝：在这两组中因子设置的唯一差别在于这对每个队列的排队规则不同(因子 6)。因此，看起来这个因子相当的不重要。
- 对某个特定的设计点，当 5 个响应的均值(大圆点)大的时候，5 个响应(小圆点)的方差就较大，这发生在“总均值”水平线上。特别是设计点 1(所有的因子都处于“－”水平)和点 64(所有的因子都在“＋”水平)的方差分别为 0.413 和 0.034。因此，对于这个问题，我们看到响应的方差在 64 个设计点上并不是常数，而这是方差分析的一个基本假设。

虽然上面的观察十分有价值，我们应当正式地证实并且还要量化这些效用。表 12.9 [$E(e_j)$为因子 j 的期望主效用]和表 12.10[$E(e_{j_1j_2})$为因子 j_1 和因子 j_2 的期望交互效用]分别给出了 6 个期望主效用，以及 15 个期望交互效用(单位：分钟)的 90%置信区间，而在图 12.4 中我们将这些置信区间画了出来。(我们并没有使用邦费罗尼不等式调整这些置信水平，因为总共有 21 个置信区间。)尽管我们也计算了 3 种方式及 3 种以上方式(包括 6 个)的交互效用，我们并没有在这里给出，因为它们非常接近 0。实际上，我们从表 12.10 以及图 12.4 中看到两种方式交互效用的幅度或者统计不显著(11 比 15)，或者很小。确实，无重要的交互效用使得可以直接解释主效用估计，接下来我们将讨论这个问题。

表 12.9 期望主效用的 90%置信区间(单位：分钟)，小型工厂模型

期望主效用	90%置信区间
$E(e_1)$	-0.17 ± 0.03
$E(e_2)$	-1.83 ± 0.29
$E(e_3)$	-0.07 ± 0.10
$E(e_4)$	-0.20 ± 0.09
$E(e_5)$	-0.23 ± 0.11
$E(e_6)$	-0.01 ± 0.05

表 12.10 期望交互效用的 90% 置信区间(单位：分钟)，小型工厂模型

期望交互效用	90%置信区间	期望交互效用	90%置信区间
$E(e_{12})$	0.008±0.013	$E(e_{26})$	−0.001±0.047
$E(e_{13})$	0.012±0.015	$E(e_{34})$	0.029±0.054
$E(e_{14})$	−0.001±0.008	$E(e_{35})$	−0.040±0.017
$E(e_{15})$	0.003±0.007	$E(e_{36})$	−0.001±0.003
$E(e_{16})$	−0.004±0.001	$E(e_{45})$	−0.004±0.008
$E(e_{23})$	−0.026±0.107	$E(e_{46})$	0.001 8±0.001 9
$E(e_{24})$	0.050±0.034	$E(e_{56})$	−0.001±0.005
$E(e_{25})$	0.103±0.056		

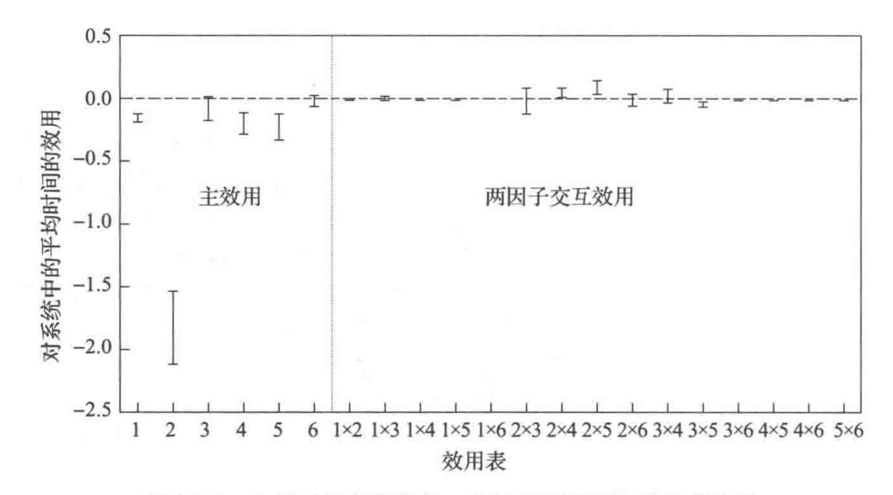


图 12.4 小型工厂实验设计：主效用和两个因子交互效用

通过减小检查时间(因子 2)可以得到在系统中的平均时间(1.83 分钟)最大程度的减小。除此以外，看起来提高质量(因子 5)，减小机器维修时间(因子 4)，或者减小加工时间(因子 1)是下一个最好的步骤。因子 3(机器正常运行时间)并不是统计显著的(表 12.9 中，其置信区间包含 0)，而且幅度也小，因此精力不应当花在提高机器的可靠性上。因子 6(排队规则)也不是统计显著的，且幅度很小，因此肯定也不值得改变排队的规则。在该系统的进一步的任何分析中[如，试图搜索因子的最优值(参见第 12.5 节)]，因子 4 和因子 6 可以设为“−”水平，并且恐怕不再考虑，以减小问题的规模到只有 4 个因子。

有 4 个交互效用具有统计显著性，2×5 交互效用具有最大的幅值(0.10)。然而，这些交互效用可能不具有实际意义。

由于我们没有发现任何重要的交互效用，主效用可以看做系统内相关因子从“−”水平变化到“+”水平的、在平均时间内带来的结果(更多的讨论参考 12.4 节)。然而，当因子在任意水平而发生变换幅度有从“−”到“+”的水平变化量时，假设主效用估计是对系统响应的精确估计也是不够可靠的。例如，一个零件(因子 5)处于“−”水平为故障，且概率为 0.10，处于“+”水平的概率为 0.09，当从“−”水平跳至“+”水平时发生 −0.01 的变化。因此，由于缺少重要的交互效用，我们将因子 5 的主效用估计解释为：如果我们将这个因子从 0.10 变到 0.09 响应的变化。然而，如果从任何值开始(从 0.99 移动到 0.98，或者从 0.43 移动到 0.42)，将这个因子移动 −0.01，并认为相同的主效用估计是相应变化的精确描述可能也是不可靠的。除非冒险假设在实验中响应对因子水平范围之外的值也是线性的。我们并没有关于响应在挑选因子水平外的函数状况知识(在这个例子中，从 0.10 到 0.09)，而且将结果和结论外推至没有实验过的区域也是有问题的。

例 12.5 考虑 13.2.1 节中的捕食者-猎物模型，在此之前请读者仔细阅读该模型的相关内容。表 12.11 给出了 5 个感兴趣的因素，以及它们的缩写和"－"、"＋"水平。响应是超过 2 000 个时间点水平的狼群存活的平均数量。◀

表 12.11　捕食者-猎物模型编码

因子编号	因子描述	－	＋
1	羊群能量增益(SG)	3	5
2	羊群再生率(SR)	3	5
3	狼群再生率(WG)	15	25
4	狼群再生率(WR)	4	6
5	青草群再生时间(RT)	15	35

利用 NetLogo(2013)仿真软件包执行 2^5 析因设计以及在每个设计点进行 $n=5$ 次重复仿真(独立随机数用于 32 个设计点)。表 12.12 给出了期望主效用近 95%的置信区间。除狼群再生率以外的所有主效用都很"大"并且统计显著。

表 12.13～表 12.15 分别给出了所期望的两因子、三因子、四因子交互效用 95%的置信区间(五因子互效用不具备统计显著性)。注意 10 个两因子的交互效用中有 5 个具有统计显著性，并且某些两因子的交互效用的幅值非常大。同样注意到 1×3×5 的交互效用的幅值也非常大(例如，28.21)并且具有统计显著性。在图 12.5 所显示的 $SR=4$，$WR=5$ 的立方图中也同样可以体现出该特性。在"盒状"立方体顶点的数字为对相应的设计点进行 5 次重复仿真后得出的平均响应。通过立方图可以明显看到，该响应是由三个因子共同作用决定的。这表明相较于人们通常认为的部分析因设计，事实上，某些模型的三因子交互效用的幅值可以很大(参见第 12.3 节)。然而，虽然个别点的值稍大于 0，但四因子交互效用不具有统计显著性。

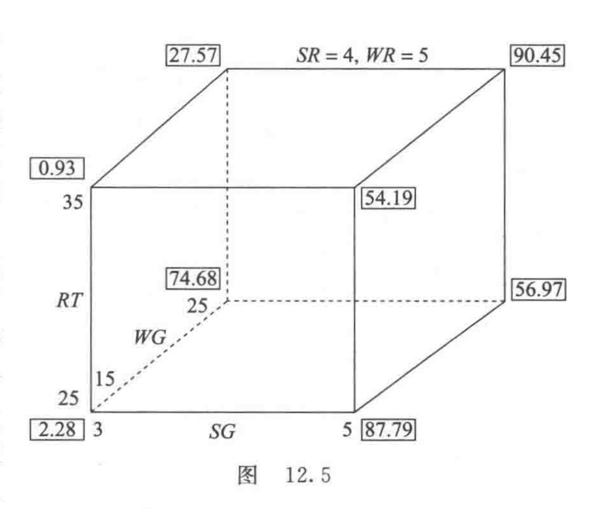


图　12.5

表 12.12　期望主效用 95%的置信区间(狼群)，捕食者-猎物模型

期望主效应	95%置信区间
$E(e_1)$	45.99±6.35
$E(e_2)$	27.33±1.52
$E(e_3)$	26.12±5.58
$E(e_4)$	−1.38±7.40
$E(e_5)$	−12.15±3.28

表 12.13　期望两因子交互效用(狼群)的 95%置信区间，捕食者-猎物模型

期望两因子交互效用	95%置信区间	期望两因子交互效用	95%置信区间
$E(e_{12})$	16.21±2.22	$E(e_{24})$	1.09±4.34
$E(e_{13})$	−23.40±6.46	$E(e_{25})$	−4.90±5.69
$E(e_{14})$	−2.28±6.20	$E(e_{34})$	−1.84±6.74
$E(e_{15})$	12.08±1.69	$E(e_{35})$	5.33±3.66
$E(e_{23})$	12.28±1.92	$E(e_{45})$	0.77±4.21

表 12.14 期望三因子交互效用(狼群)的95%置信区间，捕食者-猎物模型

期望三因子交互效用	95%置信区间	期望三因子交互效用	95%置信区间
$E(e_{123})$	2.06±2.71	$E(e_{145})$	−0.13±3.63
$E(e_{124})$	−0.56±4.98	$E(e_{234})$	0.99±4.34
$E(e_{125})$	−0.73±4.46	$E(e_{235})$	−0.81±5.56
$E(e_{134})$	−3.45±5.90	$E(e_{245})$	2.53±8.09
$E(e_{135})$	28.21±2.08	$E(e_{345})$	0.31±4.59

表 12.15 期望四因子交互效用(狼群)的95%置信区间，捕食者-猎物模型

期望四因子交互效用	95%置信区间	期望四因子交互效用	95%置信区间
$E(e_{1234})$	−0.89±4.88	$E(e_{1345})$	0.20±3.77
$E(e_{1235})$	2.78±3.78	$E(e_{2345})$	1.93±8.34
$E(e_{1245})$	1.63±7.30		

最后，注意该模型中 32 个设计点的样本方差在最小值 0.028～最大值 2 261.35 之间变化，有 80 763 种差异！

应该说明的是，即使一个因子的主效用的幅值小，这个因子也可能是重要的，下面的例子说明了这个问题。

例 12.6 假设我们有两个因子 A 与 B，“－”水平为 A^- 与 B^-，以及“＋”水平为 A^+ 与 B^+。四个设计点的响应在图 12.6 中给出。那么，因子 A 与 B 的主效用分别为：

$$e_A = \frac{(80-40)+(21-60)}{2} = 0.5$$

与

$$e_B = \frac{(60-40)+(21-80)}{2} = -19.5$$

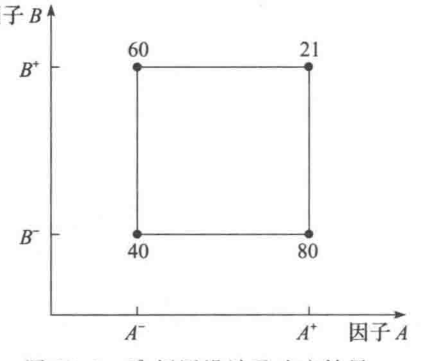


图 12.6 2^2 析因设计及响应结果

如果因子 B 处于其 B^+ 水平，那么当因子 A 从其 A^- 水平变化到 A^+ 水平时，响应减小了 39。另一方面，如果因子 B 处于其 B^- 水平，那么当因子 A 从其 A^- 水平变化到 A^+ 水平时，响应增加了 40，那么 $A\times B$ 的交互效用为：

$$e_{AB} = \frac{-39-40}{2} = -39.5$$

由于 $e_A=0.5$ 很小，初看我们可能会认为因子 A 可能根本对响应没有效用。实际上，将因子 A 从其 A^- 水平变化到 A^+ 水平对响应有很显著的影响，只是效用变化的幅度取决于因子 B 的水平。因此，我们看到一个特定因子的重要性在交互效用显著的场合能被掩盖。◀

本节中我们的例子演示了我们介绍的实验设计方法，但是读者也许发现细读这些方法(以及下节的方法)在仿真项目中的大量应用是很有趣的，这些应用诸如 Hood 和 Welch (1992)、Kang 等(2006)、Kumar 和 Nottestad(2006)、Porcaro(1996)，以及 Posadas 和 Paulo(2003)等。我们也许还要说明一下，本节讨论的 2^k 析因设计和下节的 2^{k-p} 部分析因设计已经在 AutoStat 软件工具包中实现了，这个工具包是 AutoMod 仿真软件的一个选件[参见 Applied(2013)]。同样，析因设计也可以在 ExtendSim 仿真软件中应用(参见第 3.5.2 小节)。也有几个统计软件工具包也提供了极好的实验设计能力，包括 Design Expert[Stat-Ease(2013)]、JMP[SAS(2013)]，以及 MINITAB[Minitab(2013)]。

12.3 2^{k-p}部分析因设计

例 12.4 制造模型的实验中包含六个因子，需要很多的计算量。易于设想该模型有更加复杂的版本，其中我们也许感兴趣的因子有数十个。在这种情况下，2^k 全析因设计可能很快变得无法控制。例如 $k=11$ 个因子会带来 $2^{11}=1\ 024$ 个设计点，且如果我们想每个实验点都做 $n=5$ 次重复运行(从统计学的观点来看，肯定是一个适度的样本大小)，那么这总共有 10 240 次重复运行。如果每次重复运行，比如说，需要花 1 分钟的计算机时间(对许多实际仿真模型来说是一个适度的时间)，我们可能就需要多于满满 1 周的时间不停地计算来运行这些实验。

本节首先讨论 2^{k-p}部分析因设计，它广泛应用于从所有感兴趣的因子集合中“筛选”一个重要的子集，它比全因子实验需要的计算量小：一般部分析因设计用于 15 个或者更少的因子。然后，当因子数目多得多时可能要考虑其他类型的筛选设计，我们对相关文献进行评述。对筛选设计的希望是我们能够很快地判断出哪些因子是重要的，将其他的因子固定在一个合理水平上，从而在接下来的研究中忘了它们，而将更多的注意力放在那些管事的因子上。特别是，减少因子的数目也就减小了在第 12.5 节中讨论的寻优方法的搜索空间的维数，当使用这些寻优方法时计算量上将有巨大的好处。

部分析因设计提供了一种得到主效用以及也许还有的两因子交互效用的良好估计的方法，而计算量只是 2^k 全因子设计所需计算量一部分。基本上，从所有 2^k 个可能设计点挑选出一个子集(大小为 2^{k-p})构建 2^{k-p}部分析因设计，然后只对这些被挑选的点运行仿真。由于可能的 2^k 个因子水平组合中只有 $1/2^p$ 个实际运行，如果 $p=1$，我们有时说“二分之一部分”，如果 $p=2$，说“四分之一部分”，等等。显然，从计算的角度来看，我们希望 p 大，但较大的 p 也会导致来自实验的信息较少，正如预期的那样。

从 2^k 个可能组合中挑选一个大小为 2^{k-p}的那个子集是一个十分重要问题，这个棘手的问题的深入讨论最好留给实验设计文献[参见 Montgonmery(2005，第 8 章)]。尽管如此，我们可给出一个在很多情况下可以使用的相对简单的“指南”性的步骤。

要做这件事需要我们首先讨论在 2^{k-p}部分析因设计中混淆的概念。这必然得出在这种实验中，对几个不同的效用我们可能会有完全一样的代数表达式。例如，在 2^{4-1} “二分之一部分”中，可能是主效用 e_4 和三路交互效用 e_{123}完全一样；在这种情况下，我们称因子 4 的主效用是因子 1、因子 2、因子 3 之间三路交互效用的“混淆”或者“别名”。这实际意味着 e_4 和 e_{123}的公共表达式都是 $E(e_4)+E(e_{123})$的无偏估计。我们称 e_4 和 e_{123}是相互的“别名”，并记作 $e_4=e_{123}$。现在如果我们愿意假设 $E(e_{123})=0$ 或者和 $E(e_4)$比基本可以忽略，那么 e_4 就是 $E(e_4)$的无偏(几乎是)估计。经常发生的是，多路交互效用 d_0 与主效用或者两路交互效用相比很小(正如我们在例 12.4 中已经注明的那样)，这个假设实际上成立。虽然如此，当两路交互效用相互混淆时会产生问题，例如 e_{12}和 e_{34}的表达式可能完全相同，在这种情况下该公共表达式是 $E(e_{12})+E(e_{34})$的无偏估计，从而假设两路交互效用都为 0，我们会感到不合适。更糟的是，我们可能将一个主效用与一个两路交互效用混淆，这使得在两路交互效用存在的情况下主效用估计的价值很有限(我们不能很容易地判断到底是因为两路交互效用与主效用之间的混淆、它们彼此之间的混淆，还是与更高阶之间混淆)。一般情况下，p 越大，混淆问题越普遍。

量化整体混淆程度的一个方法就是采用特定 2^{k-p}部分析因设计分辨率的概念。可以保证，如果两个效用的“路”数之和严格小于设计的分辨率，则这两个效用彼此没有混淆；为此，主效用认为是“单路”效用。最常用的部分析因设计的分辨率为Ⅲ、Ⅳ、Ⅴ(使用罗马数字记录分辨率)，它们的定义如表 12.16 所示。例如，在分辨率为Ⅴ的设计中，没有主效用与其他主效用或者与任何两路交互效用混淆($1+2<4$)，但是两路交互效用之间彼此别名($2+2=4$)。因此，假设 3 路(或多路)交互效用可以忽略，分辨率Ⅳ型设计可以

使我们得到“清晰的”主效用估计。符号上，分辨率记作角标：2_{IV}^{6-2}、2_{V}^{5-1}，等等。

表 12.16　部分析因设计中Ⅲ、Ⅳ、Ⅴ精度定义

精度	定义
Ⅲ	没有主效用与任何其他主效用混淆，但是主效用与两路交互效用混淆，且某些两路交互效用彼此可能混淆
Ⅳ	没有主效用同其他任何主效用或任何两路交互效用混淆，但两路交互效用彼此别名
Ⅴ	没有主效用或两路交互效用与其他任何主效用或两路交互效用混淆

一旦确定了因子数目 k 和想要的分辨率，我们就要求构建设计，即判断出 p 以及使用 2^k 全因子设计矩阵中哪几行。第一步是按照因子 1，2，…，$k-p$ 的顺序写出整个 2^{k-p} 因子设计矩阵，如表 12.2 和表 12.8 所示(这个基本的设计矩阵有需要的行数，即 2^{k-p} 行)。然后根据表 12.17 所示的规则[摘自 Montgomery(2005，第 305 页)]，将前 $k-p$ 列作“乘”法，来确定剩下的 p 列(对于因子 $k-p+1$，…，k)，其中“列乘”的含义是将每列的相应项一起相乘，同号相乘为“+”，异号相乘为“−”，与我们在全因子设计中计算交互效用完全一样。某个 2^{k-p} 设计所需的运行数目(设计点)也在表 12.12 中给了出来。

例如，为了构建表 12.17 中所描述的 2_{V}^{8-2} 设计($k=8$ 因子的四分之一部分设计)，我们首先按因子 1，2，…，6 的顺序写出 2^6 全因子设计，然后定义因子 7 的列为 1、2、3 与 4 列的乘积，因子 8 的列为 1、2、5 与 6 列的乘积(注意，我们可以将列 7 或列 8 的任何一列或者两列的符号反号，做法是在表 12.2 的“±”说明中取“−”；这个灵活性在仿真中证明十分有用，例如，如果总是取“+”号可能会导致建立的配置模型运行开销大)。然后，因子 j 的主效用计算同全因子设计中一样：将因子 j 的列的符号赋给响应列的对应数字，并将其求和，对其作除法，然而，除数是 2^{k-p-1}，而不是 2^{k-1}。交互效用的计算也和前面一样：将所包括的因子的列一起相乘，并将得到的符号赋给响应列，将其求和，然后除以 2^{k-p-1}；当计算交互效用时，我们应当记住，由于设计分辨率的限制，并不是所有交互效用的混淆都是清晰的。

表 12.17　2^{k-p} 部分析因设计构建规则

运行次数	因子(k)						
	3	4	5	6	7	8	9
4	2_{III}^{3-1} 3=±12						
8		2_{IV}^{4-1} 4=±123	2_{III}^{5-2} 4=±12 5=±13	2_{III}^{6-3} 4=±12 5=±13 6=±23	2_{III}^{7-4} 4=±12 5=±13 6=±23 7=±123		
16			2_{V}^{5-1} 5=±1 234	2_{IV}^{6-2} 5=±123 6=±234	2_{IV}^{7-3} 5=±123 6=±234 7=±134	2_{IV}^{8-4} 5=±234 6=±134 7=±123 8=±124	2_{III}^{9-5} 5=±123 6=±234 7=±134 8=±124 9=±1 234
32				2_{VI}^{6-1} 6=±12 345	2_{IV}^{7-2} 6=±1 234 7=±1 245	2_{IV}^{8-3} 6=±123 7=±124 8=±2 345	2_{IV}^{9-3} 6=±2 345 7=±1 345 8=±1 245 9=±1 235

（续）

运行次数	因子(k)						
	3	4	5	6	7	8	9
64				·	2_{VII}^{7-1} 7=±123 456	2_{V}^{8-2} 7=±1 234 8=±1 256	2_{IV}^{9-3} 7=±1 234 8=±1 356 9=±3 456
128							2_{VI}^{9-2} 8=±13 467 9=±23 567

在我们给出应用部分析因设计的例子前，给出几个说明是适宜的。

(1) 本步骤定义的设计实际上是指定的分辨率，这不是直接看得出来的，有兴趣的读者可以参考 Montgomery(2013，第 8 章)。那么，我们根据定义，譬如说，可以先看看表 12.17 中 2_{V}^{8-2} 设计的原因(因子 7 和因子 8 两者都取选项“+”)。如果我们构造了这个设计，然后决定计算四因子交互效用 e_{1234}，显然我们会得到与 e_7 一样的公式，因为这与因子 7 的那一列的定义完全一样。因此，因子 7 的主效用与四路交互效用混淆，但是不与三路交互效用混淆，因为在设计矩阵的主效用列的定义中使用了四个单独的因子。所有这些都同分辨率 V 设计的定义一致。

(2) 对于特定的 k，可能不存在所要求的分辨率的设计。例如，对于 $k=6$，没有分辨率 V 设计。然而，在这种情况下，我们可以使用 2_{VI}^{6-1} 设计($p=1$，分辨率 VI)，它需要 32 个设计点[参见表 12.17]。还要注意，在 Montgomery 中表 12.17 扩展到 $k=15$ 个因子。

例 12.7 使用表 12.17 中的 2_{VI}^{6-1} 设计，重新考虑例 12.4 中的小型工厂模型，因子 6 都取选项“+”。表 12.18 给出了得到的设计矩阵。一个主效应并不是其他任何主效应或者任何两路、三路或者四路交互效用的别名，一个两路交互效用并不是其他任何两路交互效用或者四路交互效用的别名。◀

表 12.18 2_{IV}^{6-2} 设计矩阵

设计点	因子					
	1	2	3	4	5	6
1	−	−	−	−	−	−
2	+	−	−	−	−	+
3	−	+	−	−	−	+
4	+	+	−	−	−	−
5	−	−	+	−	−	+
6	+	−	+	−	−	−
7	−	+	+	−	−	−
8	+	+	+	−	−	+
9	−	−	−	+	−	+
10	+	−	−	+	−	−
11	−	+	−	+	−	−
12	+	+	−	+	−	+
13	−	−	+	+	−	−
14	+	−	+	+	−	+
15	−	+	+	+	−	+

（续）

设计点	因子					
	1	2	3	4	5	6
16	+	+	+	+	−	−
17	−	−	−	−	+	+
18	+	−	−	−	+	−
19	−	+	−	−	+	−
20	+	+	−	−	+	+
21	−	−	+	−	+	−
22	+	−	+	−	+	+
23	−	+	+	−	+	+
24	+	+	+	−	+	−
25	−	−	−	+	+	−
26	+	−	−	+	+	+
27	−	+	−	+	+	+
28	+	+	−	+	+	−
29	−	−	+	+	+	+
30	+	−	+	+	+	−
31	−	+	+	+	+	−
32	+	+	+	+	+	+

由于在例 12.4 中我们已经为这些 32 位设计点做了 5 次重复运行，我们使用同样的响应计算基于 2_{VI}^{6-1} 设计的主效用，以及两路交互效用的估计。特别是，假设三因子交互效用忽略不计，在表 12.19 中我们给出了期望主效用近似 90％的置信区间。这些估计与在表 12.9 中给出它们的全因子设计的对应项到小数点后两位均相等。按照例 12.9 的全因子设计的结果，发现其三因子交互效用非常小，这种紧密吻合并不令人奇怪(对于例 12.4 中分析得到的最大的五路交互效用的大小为 0.000 9)。表 12.20 中给出了期望两因子交互效用近似 90％的置信区间。这些估计值与表 12.20 中析因设计的估计值非常相似(对于例 12.4 中分析得到的最大的四路交互效用的大小为 0.002)。因此，我们从顺序组合的 32 次运行分辨率Ⅳ的部分析因设计中可以得到的结论与我们在 2^6 全析因设计 64 个设计点中得到的一样。◀

表 12.19　2_{VI}^{6-1} 部分析因设计主效用(分钟)的 90％置信区间，小型工厂模型

期望主效用	90％置信区间	期望主效用	90％置信区间
$E(e_1)$	−0.17±0.03	$E(e_4)$	−0.20±0.09
$E(e_2)$	−1.83±0.29	$E(e_5)$	−0.23±0.11
$E(e_3)$	−0.07±0.10	$E(e_6)$	−0.01±0.05

表 12.20　2_{VI}^{6-1} 部分析因设计交互效用(分钟)的 90％置信区间，小型工厂模型

期望交互效用	90％置信区间	期望交互效用	90％置信区间
$E(e_{12})$	0.008±0.013	$E(e_{26})$	−0.001±0.051
$E(e_{13})$	0.012±0.015	$E(e_{34})$	0.028±0.054
$E(e_{14})$	−0.001±0.008	$E(e_{35})$	−0.040±0.017
$E(e_{15})$	0.003±0.007	$E(e_{36})$	−0.002±0.004
$E(e_{16})$	−0.001±0.009	$E(e_{45})$	−0.005±0.008
$E(e_{23})$	−0.026±0.107	$E(e_{46})$	0.000±0.003
$E(e_{24})$	0.050±0.034	$E(e_{56})$	0.000±0.009
$E(e_{25})$	0.103±0.056	—	—

当我们使用部分析因设计而不是全析因设计并且刚好有一个重要的三因子交互效用出现时，接下来的例子表明事情没有必要完全解决的非常好。

例 12.8 再次考虑例 12.5 中的捕食者-猎物模型，其中我们执行了 2^5 析因设计。回顾表 12.15 所示的四因子交互效用同零值有些出入，尽管它们不具有统计意义。现在我们利用例 12.5 所示的响应数据执行一个 2_V^{5-1} 部分析因设计(参见表 12.17)。现在主效用和四因子交互效用混合在一起，并且两因子交互效用和三因子交互效用混合在一起。表 12.21 的第 2 列给出了有偏主效用估计。第 3 列为析因设计的无偏主效用估计(表 12.12)。最终，第 4 列为析因设计的相关四因子交互效用的无偏估计(表 12.15)(析因设计的主效用和四因子交互效用是无偏的)。例如，第 4 列第 2 行的 1.93 是 $E(e_{2345})$ 的无偏估计。注意对于表 12.21 所示的一个特别的数据行，第 3 列和第 4 列数据相加之和等于第 2 列中的数据(正如期望的那样)。因此，由于非零四因子的交互效用，分辨率 V 部分析因设计得出了主效用“轻微”的有偏估计[但是，$E(e_4)$ 的估计实际上是一个错误的信号]。然而，这确实辨别出了因子 1、因子 2、因子 3 是最重要的。

表 12.21 析因设计主效用点估计与部分析因设计点估计的比较，捕食者-猎物模型

期望主效用	部分析因设计点估计(偏置)	析因设计点估计(表 12.12)	相关四因子交互效用点估计(表 12.15)
$E(e_1)$	47.92	45.99	1.93
$E(e_2)$	27.53	27.33	0.20
$E(e_3)$	27.75	26.12	1.63
$E(e_4)$	1.40	−1.38	2.78
$E(e_5)$	−13.04	−12.15	−0.89

另一方面，分辨率 V 部分析因设计对于 $E(e_{24})$ 的估计值为 29.30，尽管析因设计的无偏估计为 1.09(表 12.13)！这是因为析因设计的 $E(e_{135})$ 的无偏估计为 28.21(表 12.14)。

◀

正如我们刚刚在例 12.7 和例 12.8 中看到的，部分析因设计可能是筛选 k 个因子中子集的一个十分有效的工具，该子集是仿真模型的“驱动器”，与全因子设计相比，减少了计算量[同样给出了 6 个或更多因子最小的运行分辨率 V 设计，就可以减少进一步的计算需求，参见 Montgomery(2013，100-102 页)]。回到本节开头的 11 个因子的例子，我们可以，例如，构建一个 2_V^{11-4} 设计，该设计可给出主效用，以及两因子交互效用的无偏估计。这也许只需要 128 个设计点，而不是 2048 个设计点。使用之前每次运行为 1 分钟的方案，我们重复运行 $n=5$ 次，可能花 5 小时 40 分钟，而不是花全设计所需要的整整一周的时间。

偶尔我们的仿真模型可能有很多因子，必须判断出哪些对响应有影响。如果标准的部分析因设计在计算上不可行，一个不同的方法就是试图减小因子 k 的(有效)数目，这种努力可能是相当有效的，因为因子效用研究的计算量需求对 2^k 和 2^{k-p} 设计来说是按 k 指数增长的。做到这一点的一种方法恐怕就是使用成组筛选设计，例如*序贯分支法*[参见 Bettonvil 和 Kleijnen(1997)，Cheng(1997)，Kleijnen 等(2006)，Sanchez 等(2009)，Wang 等(2006，2010)]或者*序贯因子设计法*[参见 Shen 和 Wan(2009)以及 Shen 等(2010)]。对于序贯分支法，开始时所有的因子都在一个组里，对其进行测试，观察该因子组对响应是否有显著效用。如果这个组是重要的，那么它将被分解为两个子组，并测试每个子组的显著性。该步骤按照这种方式继续，每个不重要的子组以后不再考虑，而每个重要的组分为两个更小的组。最终，未在被丢弃的组中的所有的因子都一个个地进行显著度测试。为了避免在一个组中的因子的主效用相互抵消，序贯分支法假设，每个因子的“−”水平和“+”水平的选择要使得期望响应在“+”水平大于“−”水

平。以这种方式选择因子水平的能力要求效用的符号(不是幅值)已知。其他的筛选方法在 Campolongo(2000)、Dean 和 Lewis(2006)、Trocine 和 Malone(2001)中进行了综述。当因子数量很多并且具有显著意义的因子所占的百分比相对较小时，筛选设计是最有效的。

本节到此为止所介绍的方法可以应用在物理实验中，当然也可以用在仿真实验中。然而在动态仿真中，有其他对于静态模型或者物理实验来说是不可能的方法。这种概念的一种最初是由 Schruben 和 Cogliano(1987)开发的，它是在做仿真时对不同输入参数值按照不同的频率振荡，然后检查输出过程，看哪些输入参数的振荡频率在输出中可以被检测到。当一个重要的因子的振荡可能会在相应频率的输出中产生明显的振荡时，而不重要的因子的那些振荡将不会出现。通过这种办法，Schruben 和 Cogliano 能挑选出几个测试模型中的重要因子，其仿真次数比传统的实验设计方法所需的明显少；Sanchez 和 Schruben(1987)提供了一个详细例子，使用这种方法来识别习题 2.23 的非洲港口模型的重要因子。Jacobson、Buss 和 Schruben(1991)；Sanchez 和 Sanchez(1991)；Sargent 和 Som(1992)；Morrice 和 Schruben(1993a，1993b)；Morrice 和 Bardhan(1995)；Hazra、Morrice 和 Park(1997)对这类频域方法进行了更进一步的研究。

12.4 响应面与元模型

在第 12.2 节和第 12.3 节我们讨论了实验设计，它能够从 k 个因子中筛选出对响应有显著影响的子集。在有些情况下可能需要使用这些重要的因子并建立一个仿真以如何将一组特定的输入因子值转换为输出响应的元模型。该元模型通常是一阶或两阶回归方程的形式，能用于对感兴趣的其他因子水平组合响应进行预测，原因是对一些模型来讲，也许没有那么多经费来对大量的系统配置进行仿真。我们也可以使用元模型来搜索一组因子值以优化(根据需要最大或最小)响应。一般性讨论元建模和响应面的书籍有 Box 和 Draper(1987)；Khuri 和 Cornell(1997)；Myers 和 Montgomery(2002)。Barton(1998)、Barton 和 Meckesheimer(2006)、Donohue 等(1992，1993a，1993b，1995)、Friedman(1996)、Irizarry 等(2003)、Van Beers 和 Klerjnen(2004)等讨论了仿真环境下的元建模。Grier 等(1997)、Hood 和 Welch(1993)、McAllister 等(2001)、Shaw 等(1994)，Watson 等(1998)，以及 Zeimer 与 Tew(1995)给出了仿真中元建模的应用。

本章的剩余部分安排如下。第 12.4.1 小节介绍基于回归的元模型，随后，将其应用到了之前讨论过的库存模型中。第 12.4.2 小节利用中心复合设计开发了捕食者-猎物模型的元模型和二次回归模型。最后，第 12.4.3 小节讨论了空间填充设计和克里金(Kriging)方法，后者可以在构建元模型时用来替代回归分析。这些方法也同样用于开发捕食者-猎物模型的元模型，并比较了回归方法和克里金方法。

12.4.1 库存模型的介绍与分析

我们元建模的讨论将基于例 12.2 和例 12.3 中的库存模型，发现该模型的所有效用都是统计显著的。令 $E[R(s,d)]$表示重新订货点的特定值 s 以及其差值 d 的月期望平均成本。那么在 2^2 因子设计中，我们实际上假设了 $E[R(s,d)]$可以通过如下一阶回归模型表示[Montgomery(2005，第 10 章)]：

$$E[R(s,d)] = \beta_0 + \beta_s x_s + \beta_d x_d + \beta_{sd} x_s x_d \tag{12.3}$$

其中，β_0、β_s、β_d、β_{sd} 为系数；x_s 和 x_d 分别为我们这里定义的因子的编码变量。

特别是，令 $\bar{s}$、$\bar{d}$ 为表 12.3 所示的 s、d(称为因子的自然变量)的平均值，那么 $\bar{s}=40$，$\bar{d}=30$。此外，令 Δs、Δd 分别为 s、d 取“－”水平和“＋”水平时的差值，那么 $\Delta s=40$ 与 $\Delta d=40$。而 s 和 d 的编码变量定义为：

$$x_s = \frac{2(s-\bar{s})}{\Delta s} = \frac{s-40}{20} \tag{12.4}$$

$$x_d = \frac{2(d-\overline{d})}{\Delta d} = \frac{d-30}{20} \tag{12.5}$$

注意，式(12.4)将 $s=20$ 映射为 $x_s=-1$，将 $s=60$ 映射为 $x_s=+1$。同样地，式(12.5)将 $d=10$ 映射成 $x_d=-1$，将 $d=50$ 映射成 $x_d=+1$。编码变量一般用在实验设计中，因为对因子变化的响应的效用总是相对于范围[−1，+1]来度量的。

假设 $\overline{e}_s(10)$、$\overline{e}_d(10)$和 $\overline{e}_{sd}(10)$为例 12.3 运行 $n=10$ 次独立重复运行的效用估计。另外，令 $\overline{R}_F(10)$为四因子(用 F 表示)设计点且 10 次独立重复运行的平均响应[因此，$\overline{R}_F(10)$为 40 个单个响应的平均]。那么 β_0、β_s、β_d、β_{sd}的最小二乘估计[参见 Montgomery(2005，第 375-378 页)]分别为：

$$\hat{\beta}_0 = \overline{R}_F(10), \quad \hat{\beta}_s = \frac{\overline{e}_s(10)}{2}, \quad \hat{\beta}_d = \frac{\overline{e}_d(10)}{2}, \quad \hat{\beta}_{sd} = \frac{\overline{e}_{sd}(10)}{2} \tag{12.6}$$

注意，式(12.6)提供了另一种计算效用估计的途径，即通过用任意统计分析包的回归分析的能力：使用式(12.4)和式(12.5)将自然变量值转换为它们对应的编码变量值，并使用这些值和响应拟合模型式(12.3)，然后使用式(12.6)将系数相应的最小二乘估计乘 2。回归系数($\hat{\beta}_0$ 除外)是效用估计值的一半的原因是，回归系数度量 x 单位变化是对均值 $E[R(s, d)]$的效用，而效用估计是基于两个单位变化(从−1 到+1)。

将由式(12.6)所估计的系数代入到模型式(12.3)中，我们得到如下以编码变量 x_s 和 x_d 为自变量的拟合回归模型：

$$\hat{R}(s,d) = 136.819 + 9.237x_s - 3.009x_d + 5.123x_s x_d \tag{12.7}$$

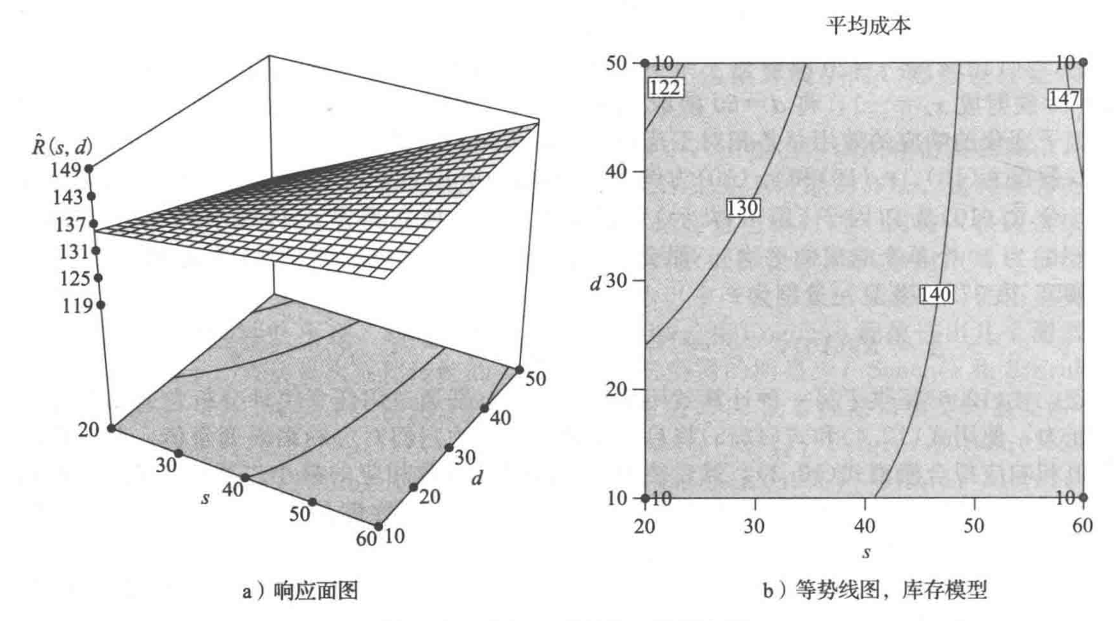
注意，系数 9.237、−3.009 和 5.123 实际上是表 12.6 中效用估计的一半(上四舍五入)。将由式(12.4)和式(12.5)给出的 x_s、x_d 代入到式(12.7)给出的模型中，我们得到如下关于自然变量 s 和 d 的等价回归模型：

$$\hat{R}(s,d) = 138.226 + 0.078s - 0.663d + 0.013sd \tag{12.8}$$

式(12.8)是仿真如何将输入参数 s 和 d 转换为输出响应$\hat{R}(s, d)$的模型，并称它为元模型(仿真模型的模型)。我们将式(12.8)给出的$\hat{R}(s, d)$绘在图 12.7a 中，该图是使用 Design-Expert 实验设计软件生成的[参见 Stat-Ease(2013)]，称为响应面，由于式(12.8)有交互效用项，该图是一个“扭曲平面”。注意，如果式(12.8)中的 sd 的系数为 0，那么 s 对响应的效用就不依赖于 d，反过来也是一样的，即 s 和 d 之间无交互效用。在图 12.7b 中我们给出了响应面的等势线图，图中沿特定的势位线的所有的(s, d)点将产生几乎相同的平均响应值。表 12.3 所示的设计点用大圆点表示，圆点旁边的值是对应的重复运行的次数(这里是 10)。

元模型式(12.8)可以看做整个仿真模型的响应面的代理模型；我们需要的就是使用小型计算器或者电子表格来计算任何感兴趣的(s, d)点对的响应。然而我们必须记住，式(12.8)只是对实际仿真的一种近似，因而可能是不精确的，特别是，当远离那些提供构建该模型的数据的 s 与 d 的值时更是如此。一个元模型，说到底，本身只是一个模型，因而相对于仿真模型来说，它可能有效，也可能无效。

近似模型式(12.8)相对实际仿真模型的精确程度如何？为了对此有一个概念上的认识，我们对 $s=0$，5，10，…，100 和 $d=5$，10，15，…，100 的全部 420 种组合进行了 $n=10$次独立重复运行。图 12.8a 给出了基于整个 10 次重复运行的仿真产生的值 $R(s, d)$的平均的响应面图。对应的等势线图在图 12.8b 中给出，例 12.3 的 4 个设计点标为大点。如果我们将图 12.7b 所示的等势线图与图 12.8b 所示的等势线图的相应区域进行比较，可以看到式(12.8)给出的元模型在四个设计点附近与“真”的响应面有较好的近似，在这四个点实际收集了数据。然而，在数据收集区域的内部，它并未提供一个合适表达。由图 12.8b还看到，最小月平均成本发生在＄110 和＄120 附近，对应的 s 大概是 25，d 在 35 和 40 之间(s 在 60 和 65 之间)。

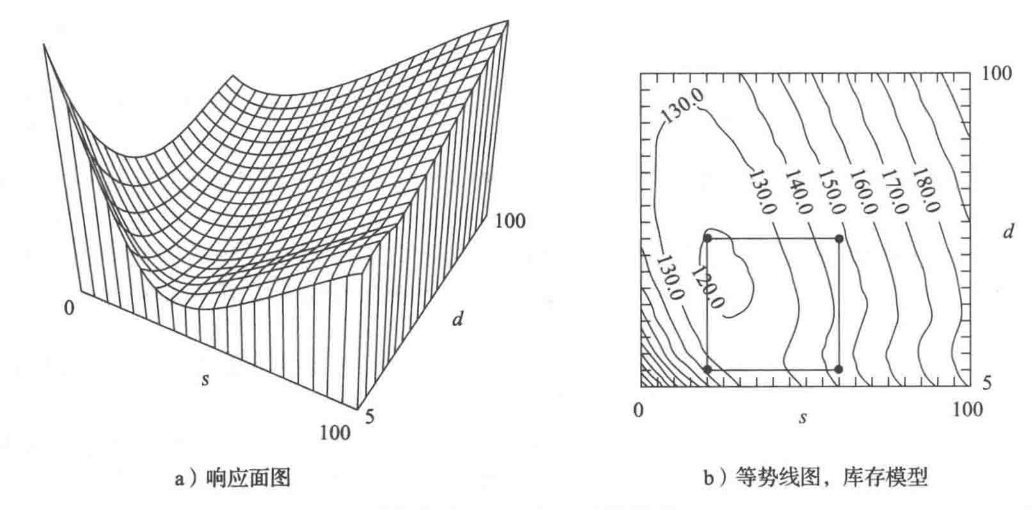


a）响应面图

b）等势线图，库存模型

图 12.7　2^2 析因设计的一阶元模型

为了产生图 12.8a 所示的响应面的数据，我们必须进行 4 200 次独立的仿真重复运行，对于大规模模型，实际中一般不可能这样做。的确，某些仿真的一次重复运行在功能强大的计算机上也要执行若干小时。



a）响应面图

b）等势线图，库存模型

图 12.8　库存模型直接仿真

式(12.3)给出的回归模型假设响应是因子效应的线性函数。然而在一些情况下，仿真模型的相应可能用下面的二次(二阶)回归模型来表示更好：

$$E[R(s,d)] = \beta_0 + \beta_s x_s + \beta_d x_d + \beta_{sd} x_s x_d + \beta_{ss} x_s^2 + \beta_{dd} x_d^2 \qquad (12.9)$$

为了确定式(12.3)给出的一阶模型是否实际上是仿真模型响应面的一个好的近似(因为类似图 12.8a 和图 12.8b 所示的图在实际应用中不能得到)或者确定是否有必要使用式(12.9)的二阶拟合模型，我们在中心点(用 C 表示)$x_s=0$、$x_d=0$(或等价地 $s=40$ 和 $d=30$)做 10 次独立重复仿真，得到平均响应 $\overline{R}_C(10)=122.95$。将 $x_s=0$、$x_d=0$ 代入到式(12.7)中，得到 $\overline{R}_F(10)=136.82$，这是一阶模型在中心点的预测响应。因此，我们得到

差值$\overline{R}_C(10)-\overline{R}_F(10)=13.87$，图 12.9 展示了中心点的两个平均值的关系，看起来还比较大[如果式(12.7)与数据收集区域内部的平均仿真响应近似得好，那么 $\overline{R}_C(10)$应当和式(12.7)定义的平面很接近]。

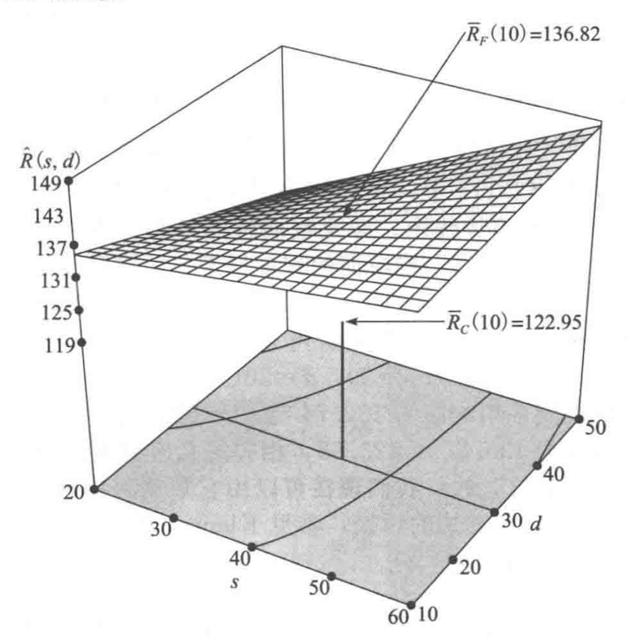


图 12.9　2^2 析因设计的一阶元模型的响应面图以及中心点的平均仿真响应，库存模型

为了看看差值 13.87 是否是统计显著的，我们构建 $E(R_F)-E(R_C)$ 90%置信区间[使用式(10.1)]，得到 13.87 ± 1.47。由于置信区间不包括 0，该差值确实是统计显著的。因此看来，它是二次(或更高阶)曲面，应该考虑式(12.9)给出的二阶模型。

可惜的是，我们无法唯一估计式(12.9)的 6 个待定系数，因为我们只从 5 个独立设计点搜集数据(即 2^2 析因设计的四个，以及中心点 1 个)。因此，我们必须在现有 5 点的基础上再添加 4 个“轴点”。这样得到的设计称为中心复合设计(central composite design，CCD)，将用来拟合二阶模型，图 12.10 给出了示意图。注意，除中心点外，其余点都位于以原点为圆心的圆上，半径为$\sqrt{2}$。具有这种性质的实验设计称为可旋转设计，含义是与设计中心距离相同的所有点，其预测响应的方差都是相同的。对于 4 个轴点，每个都做 $n=10$ 次独立的仿真重复运行，结果如表 12.22 所示(CCD 的所有 9 个设计点都是独立仿真的)。根据所有 9 个设计点的数据，可利用 Design-Expert 软件得到下面编码变量的拟合二阶模型：

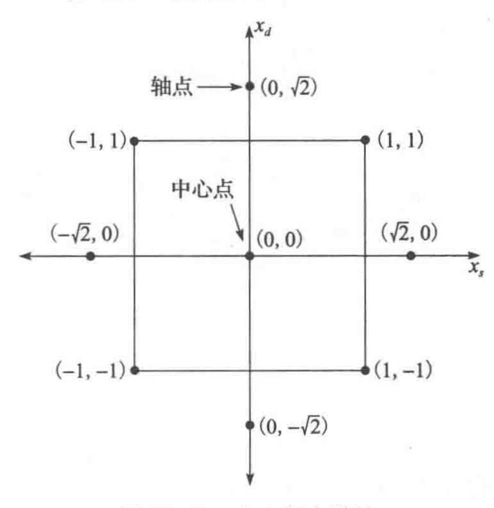


图 12.10　中心复合设计

$$\hat{R}(s,d)=122.848+8.268x_s-0.973x_d+4.627x_sx_d+9.368x_s^2+3.510x_d^2 \tag{12.10}$$

自然变量的等价二阶模型是：

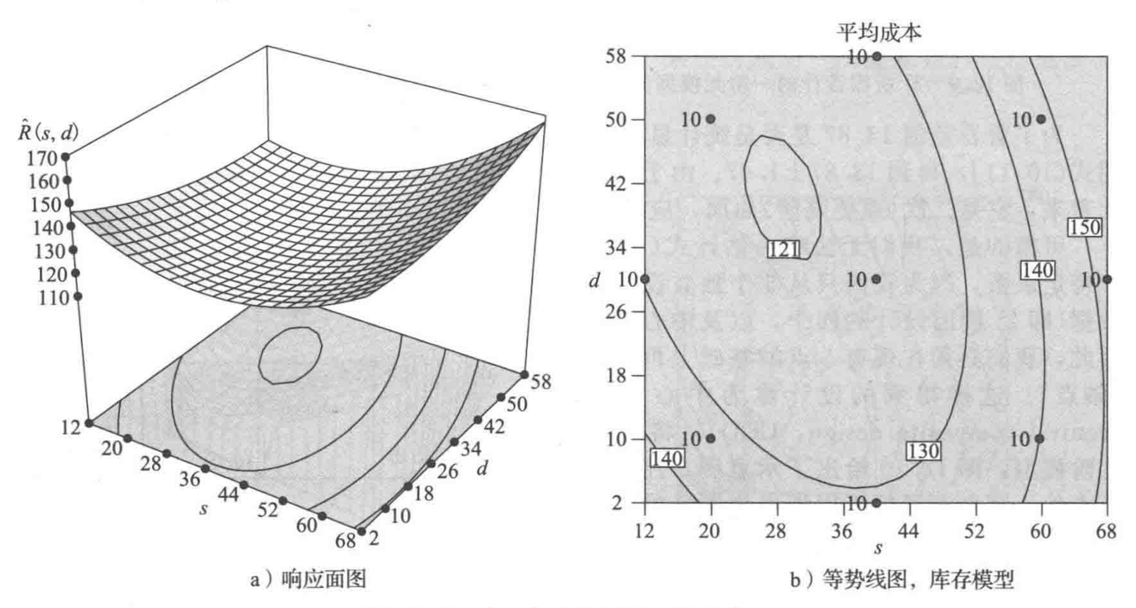
$$R(s,d)=167.020-1.807s-1.038d+0.012sd+0.023s^2+0.009d^2 \tag{12.11}$$

表 12.22 4 个轴点的响应的样本均值，库存模型

轴点(自然变量)	样本均值
$s=12$，$d=30$	129.65
$s=68$，$d=30$	150.35
$s=40$，$d=2$	125.55
$s=40$，$d=58$	130.48

将 $x_s=0$ 与 $x_d=0$ 代入式(12.10)，可得到 122.85，这非常接近中心点的平均仿真响应 $\overline{R}_C(10)=122.95$。在图 12.11a 中，我们给出了由式(12.11)定义的响应面图，而在图 12.11b 中我们给出了相应的等势线图。如果我们将此等势线图与图 12.8b 所示的可比区域相比，显然，二阶模型接近“真”响应面的程度好于一阶模型的。

为了进一步检查由二阶模型提供的拟合质量，我们首先计算调整系数 R^2_{adjusted}[参见，例如，Kleijnen 与 Sargent(2000)]，且得到可接受的值为 0.864。然后在新的设计点 $s=50$，$d=40$($x_s=0.5$，$x_d=0.5$)与 $s=30$，$d=20$($x_s=-0.5$，$x_d=-0.5$)进行了 10 次仿真的重复运行，分别得到平均响应为 132.74 与 119.02。将编码变量值代入式(12.10)，我们得到预测的期望响应为 130.87 与 123.58，相应的元模型误差为 1.41%与 3.83%。因为二阶元模型看起来是“有效”的，我们现在可以用它来预测我们的实验区域内的其他设计点的平均响应[对于其他确认方法的讨论，参见 Kleijnen 与 Sargent(2000)和 Kleijnen 与 Deflandre(2006)]。



a）响应面图　　b）等势线图，库存模型

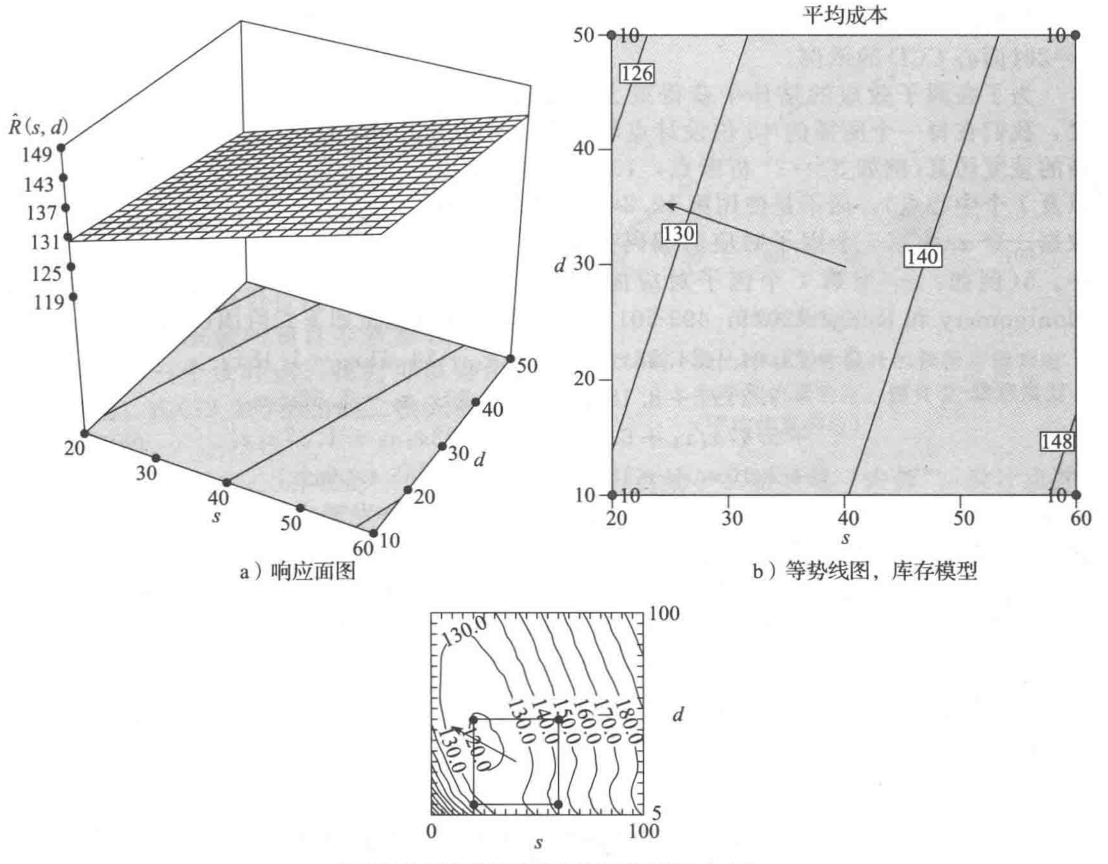
图 12.11 中心复合设计的二阶元模型

我们现在讨论经典方法以搜索在我们实验范围上给出最小平均响应的 s 和 d 值(关于优化更综合的讨论在第 12.5 节中给出)。返回例 12.3 中的 2^2 析因设计，我们对 40 个单个响应拟合一个 s 和 d 不带有交互效用项的一阶回归模型，并得到：

$$\hat{R}(s,d) = 122.857 + 0.462s - 0.150d \tag{12.12}$$

图 12.12a 和图 12.12b 给出了它的响应面图和等势线图。作为仿真的响应面的全局元模型，式(12.12)显然非常简单，但这不是用于我们希望用到的地方。相反，我们可以将式(12.12)看做期望响应面的一个局部线性近似，并可以了解到我们从设计的中心点(即 $s=40$，$d=30$)应该朝哪个方向移动，元模型高度下降得最快(由于响应表示的是成本，本例中下降是所希望的)。由微积分我们知道，最速下降的方向就是元模型的负偏微分向量

(即梯度)方向，本例中即为(−0.462，0.150)方向。图 12.12b 所示的从设计中心点出发的射线表示了该方向，和元模型的等势线垂直。根据图 12.12c 给出的“真”等势线，它似乎确实指出了一个好的移动方向。然后，我们可以沿这条线移动，在线上(或者在线附近，因为在本模型中它们必须为整数)选择 s 和 d 值，在每个点运行仿真，只要响应值还在下降就继续。当它开始上升时，就可停止，再做一次 2^2 析因设计，拟合出另一个线性模型，找到一个新搜索方向。该过程可继续，直至我们达到某个点，元模型看起来很平坦，即 s 和 d 的系数估计值都接近 0 为止。然后我们在该点拟合一个二阶元模型(例如，使用 CCD 的方法)，它需要做附加的重复运行仿真，并采用解析方法以找到给出该模型最小平均响应的 s 和 d 的值。Barton 与 Meckesheimer(2006)、Box 等(2005)以及 Myers 等(2009)详细地讨论了这个优化过程。



a) 响应面图

b) 等势线图，库存模型

c) 直接仿真的等势线图 (箭头指示搜索方向)

图 12.12 2^2 析因设计的简单的一阶元模型

上面描述的步骤显然相当粗糙、乏味，并且充满了各种出错的可能，例如，由于仿真输出的可变性，在某个点所选搜索方向完全不合适。Chang 等(2013)研发了一种自动化方法，他们将其命名为仿真优化的随机信任域响应面方法。

前面我们已经应用 Design-Expert 软件来多次绘制图形，该软件为拟合一个元模型采用非线性规划算法(即 Nelder-Mead 单纯形法)试图找到给出最小(或最大)响应的因子水平。对于库存模型，我们利用 Design-Expert 软件来找 s 和 d 值，使得由二阶元模型式(12.11)给出的平均响应 $\hat{R}(s, d)$ 最小化，同时满足约束 $12 \leqslant s \leqslant 68$，以及 $2 \leqslant d \leqslant 58$。Design-Expert 软件优化算法找到“最优”因子水平为 $s=29$，$d=40$，对应的最小平均成本为 \$120.25。回顾在图 12.8中通过查看“直接仿真”的等势线图，我们估计出最小平均

成本应在每月 \$110 和 \$120 之间某处，取 s 为 25 左右，d 在 35 和 40 之间就可能达到。

贯穿本节，我们讨论的都是有关单仿真响应。然而，很多仿真模型都有多个重要的响应，必须为每个响应都开发一个单独的元模型。但是，只有：(1)设计中包含了对于任何响应均是显著的所有因子；(2)所有的响应每次都记录下来，这样通过使用同一组的仿真运行才能拟合多个元模型。对于多重响应的进一步讨论参见 Angun 等(2009)。

12.4.2 捕食者-猎物模型

现在我们利用面心 CCD 开发捕食者-猎物模型的二次元模型，面心 CCD 是指将每个轴点都放在“立方体”的面上而不是面外[参见 Montgomer(2013，501-503 页)]。我们使用面心 CCD 而不是可旋转的 CCD 的原因是我们想要保持在原子水平方位内。图 12.13 给出了 $k=2$时面心 CCD 的示例。

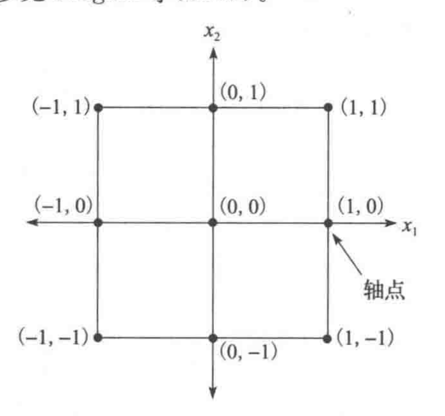


图 12.13 $k=2$ 时的面心 CCD 示例

为了在因子效应的估计中获得更大的随机统计精度，我们在每一个所需的 43 位设计点均执行 $n=30$ 次新的重复仿真(例如 $32=2^5$ 析因点，$10=2(5)$个轴点，以及 1 个中心点)，而不是使用第 12.2 节中已有的响应数据。令 x_i 为第 i 个因子对应的编码变量，$i=1, 2, \cdots, 5$(例如，x_1 为第 1 个因子对应的编码变量，SG)。使用逐步回归法[例如，参见 Montgomery 和 Runger(2011，499-501 页)]，编码变量满足如下二阶回归模型：

$$\begin{aligned}\hat{R} = & 73.04 + 24.13x_1 + 14.02x_2 + 14.61x_3 - 1.18x_4 \\ & - 7.21x_5 + 8.18x_1x_2 - 11.29x_1x_3 - 1.32x_1x_4 \\ & + 5.47x_1x_5 + 6.03x_2x_3 - 2.83x_2x_5 - 1.37x_3x_4 \\ & + 2.10x_3x_5 - 18.47x_1^2 - 9.02x_3^2 + 4.06x_4^2 \end{aligned} \tag{12.13}$$

注意，模型中没有 x_2x_4，x_4x_5，x^2 项，因为逐步回归算法发现它们不具有统计显著性(将这四个不重要的项排除在模型之外，我们得到了一个更“简约”的表示)。同时，x_4 项(对应于 WR)在模型内，很显然是因为这里使用的 30 次重复仿真可以检测到更小的因子效应。图 12.14 给出了因子 SG 和 WG 的响应面图(SR=4，WR=5，RT=30)。从图中可以注意到当 SG 趋近于 3，并且 WG 趋近于 15 时，二次元模型可以取负值。该模型中 R^2_{adjusted}的值为 0.823，该值相当大，但同时也具有误导性(参见表 12.23)。

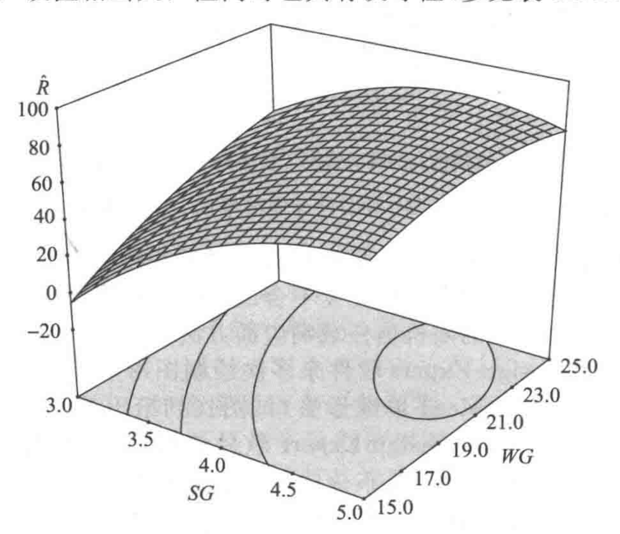


图 12.14 面心中心复合设计的响应面图，捕食者-猎物模型

表 12.23　三个新增设计点的预测和仿真平均模型对比，捕食者-猎物模型

SG	SR	WG	WR	RT	模型预测	仿真平均数	百分比错误率
4.37	3.11	15	5.89	32.37	50.87	30.94	64.41
4.47	3.21	22.37	5.16	34.47	64.76	70.35	7.95
5	3.53	19.74	4.21	30.26	72.64	82.32	11.76

图 12.15 给出了预测的期望响应和平均仿真响应的对比图。如果式(12.13)给出的模型拟合得非常完美，则绘制点均会落在一条直线上，且斜率为 1，截距为 0。

为了评估式(12.13)的拟合度，我们通过图 12.11 中给出的因子水平随机在指定的试验区域生成 20 个新增设计点。对于每一个设计点，我们都执行 $n=30$ 次独立仿真重复运行，并在此过程中计算平均仿真响应。在表 12.23 中我们给出了前三个设计点的平均仿真响应，预测期望响应源自式(12.13)，预测的百分比误差与仿真平均数有关(第 12.4.3 小节将对剩余的 17 个设计点进行讨论)。正如表所示，对于模型质量的估计非常糟糕，尤其是对于第一个设计点。我们可以得出以下结论：式(12.13)给出的二次元模型是不可用的。

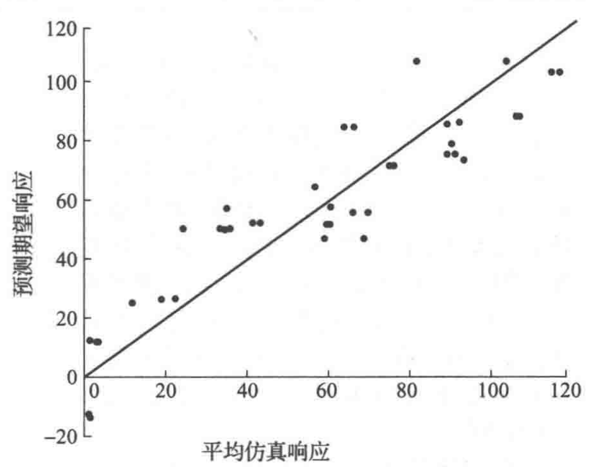


图 12.15　面心中心复合设计的预测期望响应和平均仿真响应，捕食者-猎物模型(平均仿真响应)

CCD(旋转或面心)存在的一个问题是只有中心点处于试验区域“内部”，设计点并不是“均匀分布”的。我们将在第 12.4.3 小节介绍空间填充设计概念时解决这一问题。此外，即使对于一个适当的 k 值，CCD 可能也无法承受，因为它们需要 2^k+2k+1 个设计点。例如，如果 $k=10$，则需要 $1\,045=2^{10}+2\times10+1$ 个设计点(对于 $k\geqslant5$ 的因子，通过部分析因设计有可能取代 2^k 个“核心”点)。

12.4.3　空间填充设计和克里金法

空间填充设计(SDF)是针对确定性计算机模型而研发的(例如，有限元分析或计算流体力学)，其目的是将设计点“均匀”地分布在试验区域内。空间填充设计要求因子为连续变量或者具有潜在的大量不同水平的离散变量。空间填充设计的种类很多，包括拉丁(Latin)方格设计、球状填充设计，以及均匀设计[参见 Santner 等(2003)，第 5、6 章；Hernandez 等(2012)；Montgomery(2013，第 11 章)以及 Cioppa 和 Lucas(2007)]。但是，这一节的讨论中，我们将主要关注拉丁方格设计，JMP[SAS(2013)]和 MATLAB[MathWorks(2013)]软件也支持该设计。在拉丁方格设计(LHD)中设计矩阵有 m 行、k 列，其中 m 为设计点所需的编号数(每个因子的水平)。对于一个特殊的列(因子)，m 个水平 l_1，l_2，…，l_m 为等距的并且可编码为：

$$l_1=-1$$

$$l_i=l_{i-1}+\frac{2}{m-1},\quad i=2,3,\cdots,m$$

则每一列为独立于其他所有列的随机置换。例如，假设 $k=2$，$m=5$。则表 12.24 给出了一个示例设计矩阵。选择设计点数的准则为 $m=10k$[参见 Loeppky 等(2009)]。

表 12.24　拉丁方格设计的设计矩阵，$k=2$，$m=5$

设计点	因子 1	因子 2	响应
1	20.5	0	R_1
2	1	21	R_2
3	0.5	1	R_3
4	21	0.5	R_4
5	0	20.5	R_5

再次考虑第 12.4.2 小节中的捕食者-猎物模型。我们将基于拉丁方格设计研发一款二次元模型，该模型涉及编码变量 x_1，x_2，…，x_5 的响应 R。针对具有 $m=50=10\times5$ 个设计点的拉丁方格设计，我们使用 JMP 软件生成其设计矩阵。图 12.16 给出了这些设计点到平面 (x_1, x_3) 的投影。我们对所有设计点均执行 $n=30$ 次重复仿真并在此过程中取平均值。这 50 个平均值即为构建二次元模型的数据。

图 12.16　利用拉丁方格设计得到的 50 个设计点到平面 (x_1, x_3) 的投影

利用逐步回归法，我们得到了下列拟合编码变量的二次回归模型：

$$\hat{R}=71.61+37.47x_1+13.47x_2+27.62x_3-17.20x_5+8.82x_1x_2-18.82x_1x_3$$
$$+13.43x_1x_5+5.76x_2x_3-4.11x_2x_5+10.97x_3x_5-19.30x_1^2-10.07x_3^2 \quad (12.14)$$

注意，式(12.14)给出的模型中有 13 项，对比于式(12.13)给出的模型中的 17 项，后者是基于 CCD 的。式(12.14)中 R^2_{adjusted} 的值为 0.988，同样观察 x_4(WR)项现已不在模型中了。图 12.17 给出的预测期望响应和平均仿真响应图显示除了取负值的两个设计点外，式(12.14)给出的二次元模型同响应数据拟合得非常好。但是，这两个点仅代表总设计点数量的 4%。此外，图 12.15 所示的拉丁方格设计的拟合度要比 CCD 的拟合度好很多。图 12.18给出了 SG 和 WG 的等高线图(SR=4，WR=5，RT=30)。

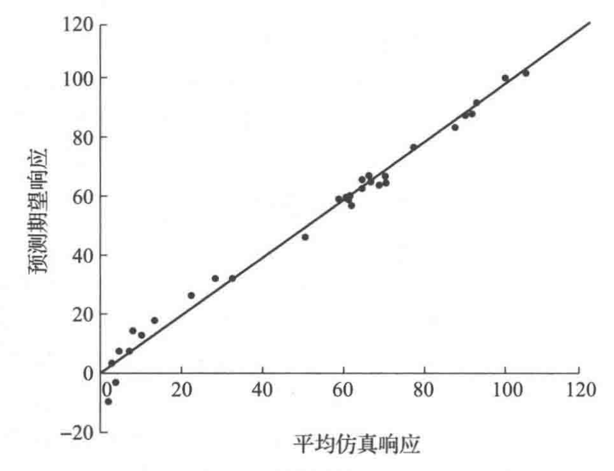


图 12.17　拉丁方格设计的预测期望响应和平均仿真响应，捕食者-猎物模型(平均仿真响应)

图 12.19 给出了利用 JMP 软件生成的二次元模型的“互作用刻画器”，该二次元模型来自于式(12.14)。因为该模型的相互作用，对改变一个特定因子响应的影响依赖于设计空间的响应点。图 12.20a中我们给出了在响应点 SG=4，SR=4，WG=20 和 RT=30 的情况下，由 JMP 软件生成的“预测刻画器”(WR 不在模型内，并将其值设为 5)。对于四个因子中的任意一个因子，垂直虚线均为其当前值，水平虚线为其预测响应(71.61)，该响应来自于式(12.14)给出的模型，在响应点处穿过两条虚线交点的实线显示了当一个因子在其范围内移动而其他因子固定不动时，预测响应是如何变化的。预测响应(例如，变化率)在响应点的导数可以用叠加三角形的大小和方向来表示。因子 SG 和 WG

所具有的三角形最大，也是位于该响应点处最重要的因子。

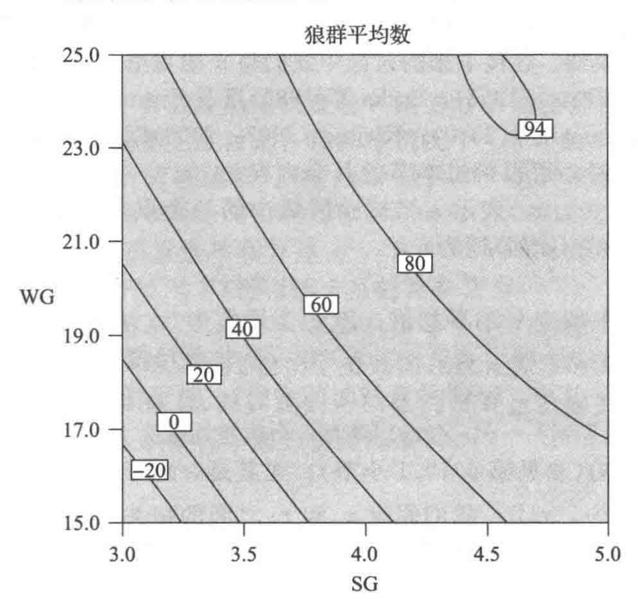


图 12.18　拉丁超立方体设计中因子 SG 和 WG 的等高线图，捕食者-猎物模型

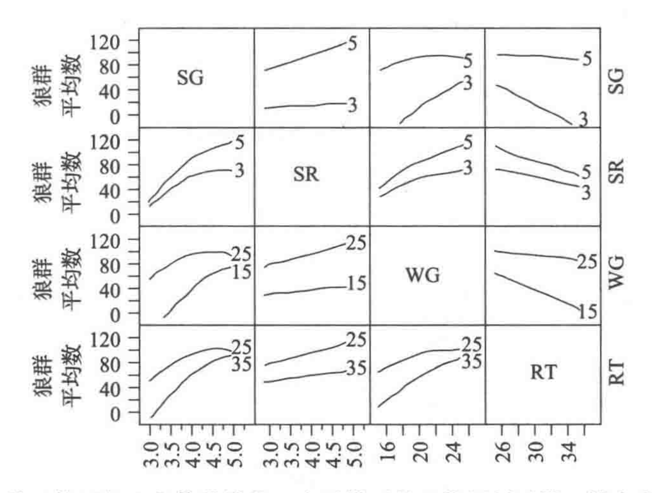


图 12.19　基于拉丁超立方体设计的二次元模型的互作用刻画器，捕食者-猎物模型

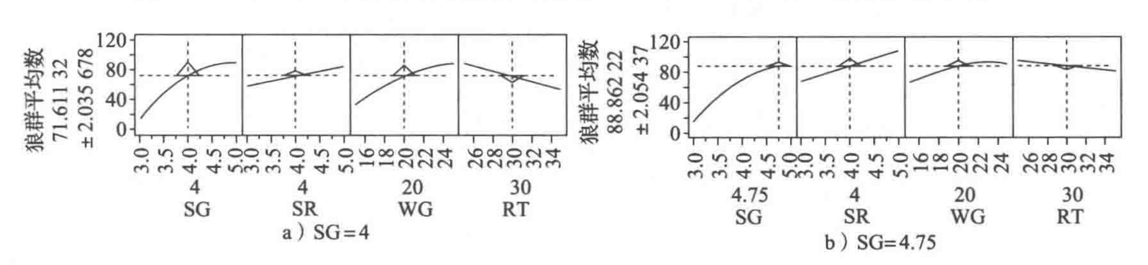


图 12.20　拉丁超立方体设计的预测刻画器

图 12.20b 给出了在响应点 SG＝4.75，SR＝4，WG＝20 和 RT＝30 处的“预测刻画器”。在此响应点处，因子 SR 和 WG 显然是最重要的。

* 低阶响应模型通常适用于为源自相对较小试验区的数据开发二次元模型，例如，这些数据是局部逼近的。另一方面，克里金法或者高斯(Gauss)过程建模一直以来都用于拟合全

局二次元模型，特别是对于确定性的计算机模型。但是，在过去的10年里，人们对于应用由荷兰蒂尔堡大学的Jack Kleijnen教授所一直倡导的克里金响应数据产生了兴趣，该数据来自随机离散事件仿真模型。在接下来的讨论中我们将介绍克里金法的使用。关于克里金法的一般性参考文献包括Cressie(1993)，Sacks等(1989)以及Santner等(2003)，而专业的随机仿真参考文献包括Ankenman等(2010)；Kleijnen(2007，2009)以及Staum(2009)。

* 在第一次阅读时，可以跳过本节的其余内容。

令 $\boldsymbol{x}=(x_1, x_2, \cdots, x_k)$表示$k$维试验区域中的一个点($\boldsymbol{x}$表示为一个向量)。则克里金模型或高斯过程模型(GPM)记为：

$$Y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}) \tag{12.15}$$

其中，μ为试验区域中响应Y的平均值；$Z(\boldsymbol{x})$为均值为0、方差为σ^2的高斯随机过程。

自相关矩阵$\boldsymbol{R}(\boldsymbol{\theta})$($\boldsymbol{\theta}=(\theta_1, \theta_2, \cdots, \theta_k)$为一个非负尺度参数的向量，random过程是stochastic过程的推广，其中，$k=1$并且按时间索引)。对于任何点$\boldsymbol{x}_1$，$\boldsymbol{x}_2$，…，$\boldsymbol{x}_m$($m>1$)，这意味着向量$[Z(\boldsymbol{x}_1), Z(\boldsymbol{x}_2), \cdots, Z(\boldsymbol{x}_m)]$具有一个边缘均值为0，边缘方差为$\sigma^2$的多元正态分布以及自相关矩阵$\boldsymbol{R}(\boldsymbol{\theta})$(参见第6.10.1小节)。尤其是，对于点$\boldsymbol{x}_i=(x_{i1}, x_{i2}, \cdots, x_{ik})$以及点$\boldsymbol{x}_j=(x_{j1}, x_{j2}, \cdots, x_{jk})$，我们假设$x_{is}$和$x_{js}$之间的相关系数是$\mathrm{e}^{-\theta_s(x_{is}-x_{js})^2}$的形式，该系数以较大的值$\theta_s$快速消退。我们进一步假设点$\boldsymbol{x}_i$和$\boldsymbol{x}_j$之间的相关系数$r(\boldsymbol{x}_i, \boldsymbol{x}_j)$为：

$$r(\boldsymbol{x}_i,\boldsymbol{x}_j) = \prod_{s=1}^{k} \mathrm{e}^{-\theta_s(x_{is}-x_{js})^2} = \mathrm{e}^{-\sum_{s=1}^{k}\theta_s(x_{is}-x_{js})^2}$$

为相关矩阵$\boldsymbol{R}(\boldsymbol{\theta})$的第$(i, j)$项。注意，$x_i$和$x_j$相隔越远，相关系数$r(\boldsymbol{x}_i, \boldsymbol{x}_j)$越小。

假设我们针对m个设计点$\boldsymbol{x}_1$，$\boldsymbol{x}_2$，…，$\boldsymbol{x}_m$运行仿真，并且令$\boldsymbol{Y}^{\mathrm{T}}=[Y_1, Y_2, \cdots, Y_m]$为相应的响应。对于一个新增设计点$\boldsymbol{x}_*$，$Y(\boldsymbol{x}_*)$的预测值为：

$$\hat{Y}(\boldsymbol{x}_*) = \hat{\mu} + \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{x}_*)\boldsymbol{R}^{\mathrm{T}}(\hat{\boldsymbol{\theta}})(\boldsymbol{Y} - \boldsymbol{1}\hat{\mu}) \tag{12.16}$$

其中，$\hat{\mu}$和$\hat{\boldsymbol{\theta}}$为从$\boldsymbol{Y}^{\mathrm{T}}$中计算得来的$\mu$和$\boldsymbol{\theta}$的最大似然估计[参见Santner等(2003，65-66页)]，$\boldsymbol{R}^{\mathrm{T}}(\hat{\boldsymbol{\theta}})$为矩阵$\boldsymbol{R}(\hat{\boldsymbol{\theta}})$的转置，$\boldsymbol{1}$为1s的一个$m\times1$向量，且

$$\boldsymbol{r}^{\mathrm{T}}(\boldsymbol{x}_*) = [r(\boldsymbol{x}_*,\boldsymbol{x}_1), r(\boldsymbol{x}_*,\boldsymbol{x}_2), \cdots, r(\boldsymbol{x}_*,\boldsymbol{x}_m)]$$

对于所有的m个设计点，式(12.16)给出的预测公式将包含一个模型项。它还将在点$\boldsymbol{x}_i$($i=1, 2, \cdots, m$)处精确地插入，例如，$\hat{Y}(\boldsymbol{x}_i)=Y_i$。注意，JMP和MATLAB[Lophaven等(2002)]软件将对GMP进行拟合以及提供预测。

克里金法可以被视为一种在随机过程中进行插值的方法。尤其是，可以证明：

$$\hat{Y}(\boldsymbol{x}_*) = \sum_{i=1}^{m}\lambda_i(\boldsymbol{x}_*)Y_i$$

其中，权重$\lambda_i(\boldsymbol{x}_*)$依赖于新增的设计点$\boldsymbol{x}_*$，并满足：

$$\sum_{i=1}^{m}\lambda_i(\boldsymbol{x}_*) = 1$$

克里金法是由南非的工程师克里金(Krige)在20世纪50年代研发出的，该方法是一种鉴于某些现有的地点黄金的数量来预测新的地点中黄金数量的方法。

再次考虑此节中前面所讨论的捕食者-猎物模型，其中，在$m=50$个设计点处进行仿真运行。令Y_i为在点i处进行$n=30$次重复仿真的平均响应。JMP软件通过Y_i计算出的GPM的估计参数为：

$$\hat{\mu} = 51.872, \sigma^2 = 1778.361, \quad \hat{\boldsymbol{\theta}} = [0.441, 0.050, 0.015, 0.013, 0.006]$$

令$\boldsymbol{x}_*=(\mathrm{SG}, \mathrm{SR}, \mathrm{WG}, \mathrm{WR}, \mathrm{RT})$，则预测方程的第一部分为：

$$\begin{aligned}\hat{Y}(\boldsymbol{x}_*) = 51.872 - 222.418\exp\{-[&0.441(\mathrm{SG}-4.27)^2 + 0.050(\mathrm{SR}-4.80)^2 \\ &+ 0.015(\mathrm{WG}-21.33)^2 + 0.013(\mathrm{WR}-4.73)^2 \\ &+ 0.006(\mathrm{RT}-32.96)^2]\} + \cdots\end{aligned}$$

我们没有给出该二次元模型的预测期望响应和平均仿真响应的对比图(相似于图 12.17)，因为我们知道一个先验：所有绘制点均会落在一条直线上，且斜率为 1，截距为 0[例如，$\hat{Y}(\boldsymbol{x}_i)=Y_i$]。

如果在仿真输出的过程中有大量的随机变量，则更好的做法是使用 GPM 来平滑响应数据而不是直接插入，因为基于上述讨论，在标准情况下 GPM 的本质是拟合“噪声”数据(回顾 GPM 曾用于确定性计算机模型)。在 JMP 软件中，这种平滑过程可以利用所谓的“金块参数”来完成[参见 Ankenman 等(2010)；Kleijnen(2009)以及 Staum(2009)]，相当于在式(12.15)中添加一个随机误差项和方差 $\sigma_\varepsilon{}^2$，其中 $Z(\boldsymbol{x})$和 ε 是相互独立的。我们使用 JMP 软件将带有金块参数 GPM 与 Y_i 进行拟合且估计参数为：

$$\hat{\mu}=13.314,\quad \hat{\sigma}_2=4\,469.058,\quad \text{金块}=8.231,\quad \hat{\boldsymbol{\theta}}=[0.171,0.015,0.004,0,0.002]$$

其中，“金块”为 $\sigma_\varepsilon{}^2$ 的估计。图 12.21 给出了最终的二次元模型的测期望响应与平均仿真响应对比图。比较图 12.17 和图 12.21，我们可以看到带有金块参数的 GPM 在两个负坐标值的设计点处的响应预测效果要优于回归模型的效果。

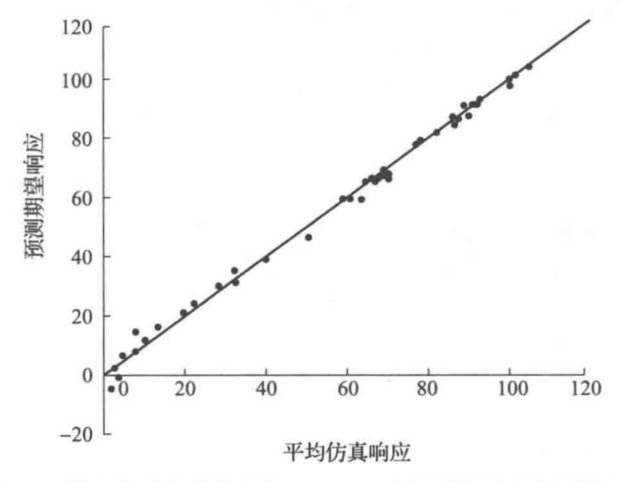


图 12.21　带有金块参数的高斯过程模型的预测期望响应和平均仿真响应，捕食者-猎物模型(平均仿真响应)

我们针对捕食者-猎物模型构建了四个不同的二次元模型：(1)基于 43 设计点 CCD 的回归模型；(2)基于 50 设计点 LHD(拉丁方格设计)的回归模型；(3)基于相同 50 设计点 LHD 的 GPM；(4)基于相同 50 设计点 LHD 的带有金块参数的 GPM。为了确定这些二次元模型的综合预测能力(例如，对其进行验证)，我们在试验区域内随机生成了 20 个新增设计点，该试验区域是由表 12.11 给出的因子水平所指定的。对于所有的这些设计点，我们进行了 $n=30$ 次重复的仿真试验，并在此过程中计算其平均响应(我们在第 12.4.2 小节中对前 3 个点进行了讨论)。在这 20 个设计点中，两个点的平均仿真响应是很小的(小于 11)，并且所有的二次元模型给出的预测响应均为负值。基于以上两点，将预测响应的值置为 0 是合理的。对于剩余的 18 个设计点，对于每一个二次元模型，我们均计算相对于平均仿真响应的预测响应的百分比误差。表 12.25 给出了 18 个设计点的均值和最大百分比误差。例如，GPM 中 18 个设计点的均值和最大错误率分别为 4.69%和 21.55%(回顾初始的 50 个设计点中的两个点，所有的四个二次元模型给出的预测响应均为负值)。

表 12.25　20 个设计点中 18 个设计点的百分比误差

误差	43 点 CCD 的回归模型	50 点 LHD 的回归模型	50 点 LHD 的 GPM	50 点 LHD 的带有“金块”参数的 GPM
均值	15.30	3.30	4.69	3.56
方差	64.41	10.77	21.55	9.83

根据表 12.25，我们可以针对捕食者-猎物模型得出如下结论：

- (空间填充)LHD 要远远优于对此仿真模型无效的 CCD。
- 带有金块参数的 GPM(平滑的)的性能优于标准的 GPM(初始设计点的精确插值)。
- 对于捕食者-猎物模型，基于 LHD 的回归二次元模型和基于 LHD 的带有金块参数的 GPM 的效果最优。

Ankenman 等(2010)开发了一个可用于随机仿真的 GPM，并允许自相关高斯随机过程在每一个设计点处的方差不同。该方法可以使用软件在 MATLAB 中实现，该软件可在 www.stochastickriging.net 网站中获得。Chen，Ankenman 和 Nelson(2010)研究了有关 GPM 的常见随机数的作用。

12.5 基于仿真的优化

分析仿真模型的终极目的可能就是寻找输入因子的组合，以优化关键输出性能指标。例如，也许有一个输出与经济有直接的重要关系，例如，利润或者成本，我们可能想对输入因子的所有可能值来使该输出最大化或者最小化。

一般情况下，涉及的输入因子可能包括离散定量的变量，如制造系统中某个工作站的机器数目，也包括连续定量的变量，如机器的平均加工时间；或者是定性的变量，如排队规则的选择。尽管在一个仿真研究中有可能要搜索可控的以及不可控的输入因子的最优值，但在大多数应用中主要关注的是作为设备设计或者运行策略的一部分可控的输入因子。

初看起来，这个优化的目标和在 10.4 节讨论的选择最优的系统这个目标似乎相当类似。然而，在那里我们假设感兴趣的备选系统配置直接给出，且备选方案的数目小(例如 20 个或更少)。这是很多仿真研究中的典型情况。但是现在我们所处的情况结构性弱得多，这里我们必须决定何种备选系统配置进行仿真，以及如何评价并比较其结果。由于我们可能要观察大量输入因子的所有组合，要仿真并比较的备选配置的数目理论上会是成千上万个。

用经典的数学优化方法来思考这个问题会有所帮助，如线性规划或者非线性规划。我们由仿真得到输出性能度量，譬如说 R，它的值依赖于输入因子譬如说 v_1，v_2，…，v_k 的值；这些输入因子是优化问题的决策变量。由于 R 是仿真的输出，它一般是带有方差的随机变量。目标就是对于所有 v_1，v_2，…，v_k 的可能组合，最大化或者最小化目标函数 $E[R(v_1, v_2, \cdots, v_k)]$。输入因子的组合可能有约束，例如下面形式的范围约束：

$$l_i \leqslant v_i \leqslant u_i$$

l_i(下限)和 u_i(上限)为常数，以及更一般的约束，可能是 p 线性约束，形式为：

$$a_{j1} v_1 + a_{j2} v_2 + \cdots + a_{jk} v_k \leqslant c_j$$

a_{ji}、c_j 为常数($j=1, 2, \cdots, p$)。例如，如果 v_1、v_2、v_3、v_4 是我们决定购买的 1、2、3、4 型机器的数量，a_{1i}为 i 类型机器的成本，c_1 为购买机器的预算，那么在选择 v_i 值的时候，我们必须满足机器预算约束：

$$a_{11} v_1 + a_{12} v_2 + a_{13} v_3 + a_{14} v_4 \leqslant c_1$$

通常情况下，如果输出 R 是我们想要搜索的最大化的利润，那么问题可正式描述为：

$$\max_{v_1, v_2, \cdots, v_k} E[R(v_1, v_2, \cdots, v_k)]$$

满足：

$$\begin{gathered} l_1 \leqslant v_1 \leqslant u_1 \\ l_2 \leqslant v_2 \leqslant u_2 \\ \vdots \\ l_k \leqslant v_k \leqslant u_k \\ a_{11} v_1 + a_{12} v_2 + \cdots + a_{1k} v_k \leqslant c_1 \end{gathered}$$

$$a_{21}v_1 + a_{22}v_2 + \cdots + a_{2k}v_k \leqslant c_2$$
$$\vdots$$
$$a_{p1}v_1 + a_{p2}v_2 + \cdots + a_{pk}v_k \leqslant c_p$$

解决这个问题在实际仿真过程中通常的确会是令人望而生畏的。首先，如在任何优化问题中一样，如果决策变量的数目(仿真中的输入因子)k 很大，那么我们是在 k 维空间中寻找一个最优点；当然，有很多使用了好几十年的数学规划方法已经用于求解这些问题。其次，在仿真中，我们计算目标函数并不能简单地将一组可能的决策变量值代入到简单的闭合表达式中，的确，必须运行整个仿真本身以得到上述表达式中的输出 R 的一个观测值。最后，在随机仿真中，由于输出的随机性，我们也不能精确地计算目标函数；改善这个问题的一种方法就是重复仿真，譬如说，对感兴趣的输入因子的一组值仿真 n 次，并用整个重复运行的 R 的平均值 $\overline{R}$ 作为目标函数在该点的估计值，n 越大，估计得越好(当然，计算量也越大)。

尽管如此，有很多最近研究是寻找仿真优化方法的，第 12.5.1 小节和第 12.5.2 小节将会提到一些。我们也看到这些方法中的一些被开发成与仿真软件(参见第 3 章)一起运行的实际优化工具包，在寻找优化目标函数的点时，智能地搜索因子空间；在第 12.5.2 小节我们讨论了这些优化包中的几个。显然，搜索最优系统配置通常是计算量很大的工作，近些年计算机硬件的发展自然对仿真优化的算法和软件两者都起了很大的帮助作用，我们期待这个趋势将继续下去。

尽管既有理论又有实际方面的挑战，哪怕找到一个近似最优的系统配置在实践中也具有潜在的巨大回报，因此，该领域的研究和活动都很多。在文献的报告中有许多有关各种方法和软件的应用，一些例子如下：

- 急诊室的运作[Fu 等(2005)]；
- 汽车制造[Spiechermann 等(2000)]；
- 生产库存系统的管理[Kapuscinski 和 Tayur(1999)]。

12.5.1　优选法

研究人员提出并开发出了很多试图通过搜索可能的输入因子组合的空间来优化仿真的不同方法。这些搜索算法在它们如何工作以及需要哪些信息(例如，是否需要估计路径上的导数)方面相差很多。然而，这些算法通常可以归类为下列方法之一：

- 元启发式算法，如遗传算法、模拟退火，以及禁忌搜索(参见第 12.5.2 小节)；
- 响应面法(参见第 12.4 节)；
- 序优化[Ho 等(2000)、Ho，Sreenivas，和 Vakili(1992)]；
- 基于梯度的算法[Fu(2006)、Glasserman(1991)、Ho 和 Cao(1991)、Rubinstein 和 Shapiro(1993)、Spall(2003)]；
- 随机搜索[Andradottir(2006)]；
- 采样路径优化[Gurkan 等(1999)、Robinson(1996)]。

然而，本节我们主要关注元启发式算法，这些算法启发式地保证其他搜索避免陷入局部最优[其他定义参见 Blum 和 Roli(2003)]。这些方法有一些实际上已经在商业仿真软件包中实现了，如在 12.5.2 节中将要讨论的那样。

仿真优化有很多好的综述，包括 Andradottir(1998)；Azadivar(1999)；Fu(2002)；Fu，Glover，和 April(2005)；Olaffson 和 Kim(2002)；Swisher 等(2004)；Tekin 和 Sabuncuoglu(2004)。讨论仿真优化新的方法学的论文有 Hong 和 Nelson(2006)、Boesel，Nelson 和 Ishii(2003)、Shi 和 Olafsson(2000)。

12.5.2　与仿真软件有接口的优选法软件包

正如在 12.5.1 节中指出的，已经提出的很多方法都是搜索优化目标函数的仿真模型

的输入组合。这些方法必须对一系列的系统配置进行仿真，用先前配置仿真的结果来为搜索可能的输入因子组合的空间给出新方向的建议，得到的是我们希望一定给出更好系统性能的配置。这样，就有一个非常实际的问题，即如何管理这一系列系统配置；对于可能需要仿真的配置数量很大的情况，显然，用人工方式运行不同的系统配置，并根据结果给出下一个配置的建议，这样做不大可行。然而，由于可以采用快速 PC 以及启发式优化搜索改进算法，现在大部分离散事件仿真软件商都将优选法软件包集成到它们的仿真软件中。

优选法软件包(简称为优化包，但应当记住，这些包不保证真的最优)的目的就是安排一系列系统配置的仿真(每个配置对应于输入决策变量或因子的一个特定的设置)，从而最终得到为配置提供一个接近最优的解。当然得到这个解使用仿真的次数要尽可能少。优化包和仿真模型的交互效用如图 12.22 所示。优化包首先指令仿真模型进行初始系统配置的一次或多次重复运行(用户可以基于系统知识选择该初始配置，或者可由优化包指定初始配置)。这些重复运行的结果(目标函数)被返回到优化包，优化包然后利用其内置的搜索算法判断是否需要仿真另一个系统配置。这个过程一直进行下去直至满足优化包的停止条件为止。

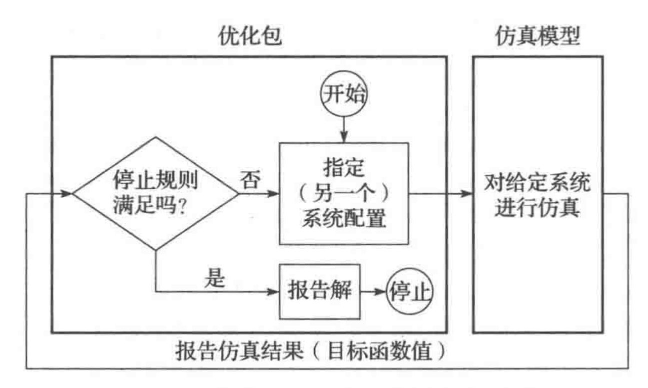


图 12.22　优化包与仿真包之间的交互效用

图 12.22 所示的操作设置确实很方便，它可以使得在所选的仿真软件环境下对优选一系列系统配置进行仿真的工作成为可能。但是，重要的是要记住并不保证结果绝对是最优的，其全部原因在第 12.5 节的开始已经讨论过了。更进一步，优化软件包需要用户指定一系列的选项和参数，且在一个给定的环境下如何做可能并不明确——然而结果恐怕将依赖于用户怎么选择这些参数。尽管有这些问题，我们觉得这些包往往会找到到一个接近最优的模型配置，或者在任何情况下，相对于分析人员以费时的、无计划的系统配置顺序方式进行“手工”寻找来说，都会找到一个看起来是上好的配置。

希望优化包具有的一些特征如下。

- 两个最重要的特征是所得到的解的质量(虽然这在实际中本质上是不可能到基准点的，因为真正的最优解是无法得知的)，以及得到解的执行时间的数量。总执行时间依赖于需要仿真的系统配置的数目以及每个配置的执行时间。后者执行时间依赖于所使用的仿真软件的速度(参见第 3.4.1 小节)以及当配置仿真完成时仿真软件多快才能得到需要返回到优化包的仿真结果。
- 在执行过程中，应当能够动态显示重要的信息，包括整个当前配置的最好的目标函数值的图，以及目标函数值，系统配置，还有 m 个最好系统配置的配置编号，其中 m 是由用户规定的。
- 决策变量的线性约束应该允许作为问题公式表示的一部分。如果能包括非线性约束，如 $v_1^2+v_1v_2\leqslant c$，那就更好了，其中，v_1 和 v_2 为决策变量，c 为常数。

- 如果能对输出随机变量定义一个约束，则会很有用；例如，人们有可能只想考虑所观测的工作站的利用率低于 0.8 的系统配置。对与输出随机变量相关的约束的特定配置的可行性检查，只有在该配置的仿真运行完成之后才能进行。
- 优化包应当包括几种停止准则，如指定数目的系统配置无改进，指定数目的系统配置已经完成，超过了指定数量的自然时间，以及枚举(如果系统配置的数目相对较少)。
- 对于 m 个最好系统配置的每一个，应当提供目标函数的期望值的置信区间。
- 有些系统配置的目标函数的估计值 $\overline{R}$ 比其他的变化大，因此，我们可能要对大方差的配置做更多的重复运行，以便目标函数的估计对于所有的配置，都有相同的精度。例如，有两台机器的工作站的仿真模型的方差一般比有三台机器的相同的工作站的仿真模型的方差大，因为在前一种情形，工作站的利用率高一些。因此，理想情况下，特定系统配置的重复运行次数应当依赖于从最初少次数重复运行计算得到方差估计；对于这个问题，可用第 9.4.1 小节的序贯程序法来完成。
- 与优化包一起使用的仿真软件在每个配置被仿真之前应该将所有随机数流的种子复位回到它们的默认值，以便促进方差缩减技术的通用随机数的使用(第 11.2 节)。
- 由于优化问题可能需要大量的执行时间，应能在联网的计算机上同时对特定配置进行所需要的重复运行。

表 12.26 列出了几个写此书时已经可用的优化软件包、它们的经销商、它们支持的仿真软件，以及所使用搜索算法。正如能看到的那样，五个包都用了不同的启发式搜索算法，包括进化策略[Back(1996)、Back 和 Schwefel(1993)以及 Schwefel(1995)]，遗传算法[Michalewicz(1996)]，神经网络[Bishop(1995)和 Haykin(1998)]，散点搜索[Glover(1999)、Laguna(2002)，以及 Laguna 与 Marti(2003a)]，模拟退火[Eglese(1990)与 Henderson 等(2003)]，以及禁忌搜索[Glover 和 Laguna(1997，2002)]。虽然我们不可能详细讨论这些工具包是如何工作的，但是我们将会对 OptQuest 和 WITNESS Optimizer 做简短的介绍，在我们下面的例子中我们将也用到他们。

表 12.26　优化软件包

软件包	销售商	支持的仿真软件产品	所用搜索算法
AutoStat	Brooks Automation	AutoMod，AutoSched	进化策略
Extend Optimizer	Imagin That，Inc	Extend	进化策略
OptQuest	Optimization Technologies	Arena，Flexsim，Micro Saint，Promodel(可选)，QUEST，SIMPROCESS，SIMUL8	散点搜索，禁忌搜索，神经网络
SimRunner2	PROMODEL 集团	MedModel，Promodel，ServiceModel	进化策略，遗传算法
WITNESS Optimizer	Lanner 集团	WITNESS	模拟退火，禁忌搜索

OptQuest[Laguna(2011)]软件包使用散点搜索实现(基于总体的元启发式算法)作为其主搜索算法，禁忌搜索和神经网络起次要作用。一个可用的停止准则是让优化算法一直运行直到完成用户指定的配置数(NC)为止。另外一个停止准则(称为自动停止)是让优化算法一直运行，直到 100 次连续的配置的目标函数值没有改进为止(更详细的情况参见下面 WITNESS Optimizer 的讨论)。也可能同时选择 NC 的值和自动停止，而且在这种情况下，优化将一直运行直到无改进的配置数目达到 NC 的 5%时为止。OptQuest 软件包还具有以下的特点：

- 对决策变量和输出随机变量具有线性约束以及非线性约束；
- 允许特定配置的重复运行次数依赖于其方差估计的选项；
- 排序选择方法统计保证 OptQuest 软件包返回的最好系统配置至少是实际被仿真的配置中最好的(参见 10.4.3 节)。

WITNESS Optimizer[参见 Lanner(2013)]软件包在主搜索程序中采用模拟退火和禁忌搜索算法，称为适应热统计模拟退火(adaptive thermostatistical simulated annealing)[参见 Debuse 等(1999)]。停止准则有两个用户自定义参数，最大配置数目(maximum number of configurations，MC)以及目标函数无改进的配置数目(CNI)。例如，假设 MC=500，CNI=100，配置 j 的目标函数目前是最好的。那么若在配置 $j+1$，$j+2$，…，$j+100$ 中没有一个目标函数值比在配置 j 的目标函数好，当运行到配置 $j+100$ 时算法终止，然而，算法也不会超过 500 个配置。也有一个所有可能的组合停止准则，这对于小型问题特别有用(参见例 12.10)。WITNESS Optimizer 软件包具有以下的特点：

- 允许决策变量具有线性约束；
- 即使指定了约束，也会为一个优化问题计算可行配置数目；
- 对于一个特定的配置，提供一种机制，以评估目标函数在重复运行到重复运行的可变性。

例 12.9 考虑例 12.2 中的库存问题，但这次有决策变量(或者因子)为 s 和 S。令 $R(s, S)$为与特定(s, S)对应的在 120 个月的计划水平上的月平均成本。假设 s 可能的取值为 0，1，…，99，而 S 可能的取值为 1，…，100。但是，S 必须比 s 大。那么我们的优化问题的正式表述如下：

$$\min_{s,S} E[R(s,S)]$$

满足范围约束

$$s \in [0,1,\cdots,99]$$
$$S \in [1,\cdots,100]$$

以及一般的线性约束

$$S-s \geqslant 1$$

因此，两个决策变量的可行组合共有 5 050=100×101/2 个。

我们使用了 OptQuest 优化包作为 Arena 仿真软件中的实现来执行优化，每个配置做 $n=10$ 次重复运行(120 个月)，停止准则为 NC=200；且 $s=20$ 与 $S=40$ 为初始配置(参见例 10.3)。OptQuest 软件包找到的最好解为 $s=27$，$S=61$，相应的月平均成本为 \$118.47，是在第 137 次配置中实现的。注意这里的 s 与 S 值与图 12.7b 所示等势线图给出的最小月平均成本的 s 与 d 值是相当一致的(在图 12.7b 中，参数 s 与 d 每次变化的增量为 5)。此外，为获得"最优"解的 137 个配置只表示了 s 与 d 的 5 050 个可行组合的 2.7%。

令 $\overline{R}_i$ 为第 i 个被仿真的配置整个 10 次重复运行的平均成本，且令 $m_i=\min\{\overline{R}_1, \overline{R}_2, \cdots, \overline{R}_i\}$(因此，$m_i$ 是 i 的一个非增函数)。在图 12.23 中我们绘制了 $i=1$，2，…，200 时的 m_i 值，从图可以看出 m_i 在 $i=19$之后减小不多。 ◀

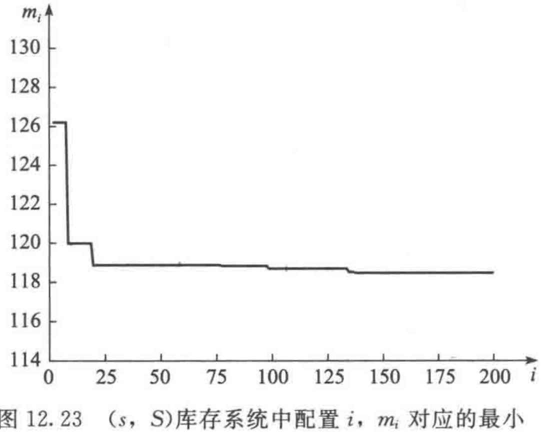


图 12.23　(s, S)库存系统中配置 i，m_i 对应的最小的月平均成本

例 12.10 考虑由 4 个工作站组成的加工线，有三个有限大小的缓冲区(队列)，以及一个不限量供应毛坯零件区，如图 12.24 所示。零件在站 1 到站 4 顺序加工(每个站都增加值)，然后离开系统。特定站的机器上的加工时间为指数分布，均值在表 12.27 中给出。在站 1、站 2、站 3 的机器完成其当前零件后，它将该零件推送到下游缓冲区，除非缓冲区的所有位置都已占满。在这种情形下，完成工作的机器变成被阻塞而不能加工另一个零

件，直到缓冲位置被释放。在站 1 的机器永远不空闲(只有忙或者被阻塞)，因为总有一个毛坯零件在等待加工。在站 2、站 3 的机器可能是忙、闲或者被阻塞，而在站 4 的机器只能是忙或者闲。◀

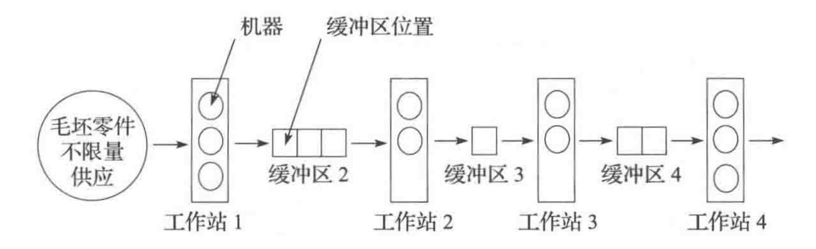


图 12.24 加工线的布局

表 12.27 4 个站每站的机器的平均加工时间，加工线

工作站	每台机器的平均加工时间(单位：小时)
1	0.333 33
2	0.500 00
3	0.200 00
4	0.250 00

这个问题有 7 个决策变量。令 $v_i(i=1, 2, 3, 4)$为站 i 的机器数目，且令 $v_i(i=5, 6, 7)$为在缓冲区 $i-3$ 处的缓冲位置数目。我们感兴趣的是，判断在 30 天周期内 $v_1 \sim v_7$ 的值为多少将实现期望利润最大化。特别是，假设加工厂商卖一个零件可以得到 \$200。进一步假定并且使用任何站的机器 30 天的成本是 \$2 500(一般说来，不同站的机器会有不同的成本)。另外，由于场地需求，使用一个缓冲存行态位置 30 天的成本是 \$1 000。假设该工厂正在考虑为每个站买 1～3 个机器(即 $i=1$、2、3、4，$l_i=1$ 且 $u_i=3$)，并且为每个缓冲区购买 1～10 个位置(即 $i=5$、6、7，$l_i=1$ 且 $u_i=10$)。因此，7 个决策变量共有 $81\,000=3^4\times10^3$ 个不同的组合。令 N(随机变量，其分布依赖于决策变量值)为 30 天生产出来的零件数目(生产量)。那么 30 天周期的利润 $R(v_1, v_2, \cdots, v_7)$为：

$$R(v_1,v_2,\cdots,v_7) = 200N - 25\,000\sum_{i=1}^{4} v_i - 1\,000\sum_{i=7}^{5} v_i$$

而我们的优化问题可以形式化表述为：

$$\max_{v_1,v_2,\cdots,v_7} E[R(v_1,v_2,\cdots,v_7)]$$

满足范围约束：

$$v_i \in \{1,2,3\}, \qquad i=1,2,3,4$$
$$v_i \in \{1,2,3\cdots10\}, \qquad i=5,6,7$$

我们使用 WITNESS Optimizer 软件包来实现优化，每个配置做 $n=5$ 次重复运行，停止准则为 MC=500，CNI=100，且 $v_i=2(i=1, 2, 3, 4)$，且 $v_i=6(i=5, 6, 7)$作为初始点。由于我们感兴趣的是加工线的稳态性能，每次重复运行长度是 40 天，前 10 天是预热期。确定 10 天的预热期已经足够，根据是对该系统的几个配置应用韦尔奇(Welch)图形算法(参见第 9.5.1小节)，10 天远大于这些配置实际所需要的预热时间(严格来讲，系统的 81 000个不同配置可能需要不同的预热期)。WITNESS Optimizer 软件包找到的最好解为 $v_1=3$，$v_2=3$，$v_3=2$，$v_4=2$，$v_5=7$，$v_6=8$，$v_7=3$，相应的平均利润为 \$578 400，是在第 124 个配置上实现的。注意这 124 个配置只表示了 7 个决策变量 81 000 个可能组合的 0.15%。

令 $\overline{R}_i$ 为第 i 个配置的整个 5 次重复运行的平均利润，且令 $\overline{M}_i=\max\{\overline{R}_1, \overline{R}_2, \cdots, \overline{R}_i\}$(因此 M_i 为 i 的非减函数)。在图 12.25 中我们绘出了 $i=1, 2, \cdots,$

150 的 M_i 的图，以及下面将要讨论的最优期望利润估计的图。

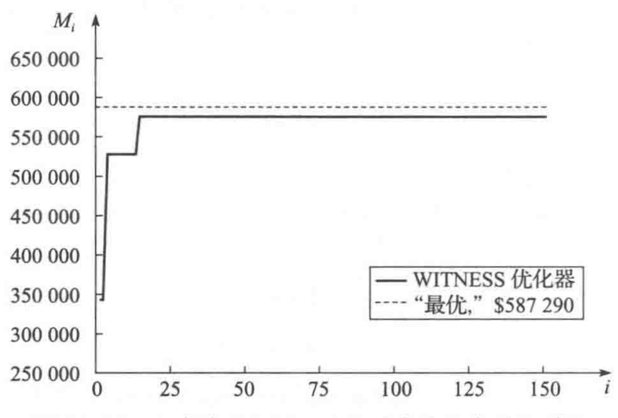


图 12.25　生产线的配置 i，M_i 对应的最大平均利润

自然要问，$578 400 与该问题真正的最优期望利润接近度如何呢？为了说明这个问题，我们使用 WITNESS Optimizer 软件包中的所有可能组合选项，对加工线 81 000 个配置的每个都进行了 $n=5$ 次重复运行，我们发现最好的 200 个配置(我们跟踪的数)的每一个为 $v_1=3$，$v_2=3$，$v_3=2$，以及 $v_4=2$。利用 $v_1 \sim v_4$ 这种设置，我们然后对 $v_5 \sim v_7$ 的 1 000 种组合的每个都进行 $n=100$ 次重复运行。这三个决策变量的最好组合结果是 $v_5=7$，$v_6=9$与 $v_7=4$，相应的平均利润为 $587 290(然而，有三种其他配置的平均利润大于 $587 000)。因此，用 WITNESS Optimizer 软件包得到的平均利润 $578 400 不同于“最优”的期望利润 $587 290，相差大约 1.5%(对每次重复运行具有“合理的”执行时间的实际仿真模型来说，这里讨论的枚举法不可行)。

例 12.11 在此考虑例 12.8 的加工线，但现在假设每台机器都随机发生故障停机。特别是，机器从忙到故障的时间(参见第 14.4.2 小节)具有均值为 10 小时的指数分布，维修机器的时间为伽马分布，均值为 0.5 小时，形状参数为 2(参见第 6.2.2 小节)。此外，机器的维修需要用一个机械师，没有机械师时停机的机器加入 FIFO 的机械师队列。每个机械师成本是加工厂的工资 $4 000 及 30 天的利润，工厂正在考虑雇佣一个还是两个机械师。

这个问题有 8 个决策变量，v_8 为要雇的机械师的数量($l_8=1$，$u_8=2$)。因此，8 个决策变量共有 162 000 个组合。我们再一次应用 WITNESS Optimizer 软件包进行优化，每个配置$n=5$次重复运行，MC=500 与 CNI=100，$v_i=2(i=1, 2, 3, 4)$，$v_i=6(i=5, 6, 7)$，$v_8=1$ 作为初始配置。找到的最好解为 $v_1=3$，$v_2=3$，$v_3=2$，$v_4=2$，$v_5=9$，$v_6=10$，$v_7=5$ 与 $v_8=2$，相应的平均利润为 $528 880，是在第 218 次配置中实现的。这里的平均利润较例 12.8 中的少，因为机器生产零件的可用时间更少了，而且必须付每个机械师 $4 000工资。◀

对例 12.9 到例 12.11 中的简单库存和加工系统来说，根据选择接近最优解的配置以及这样做的方式的有效性来看，优化软件包似乎提供了非常好的结果。然而，这是我们实验的部分结果，通过设置不同停止规则(即 NC 和 CNI)和每个配置的重复运行次数 n 得到的。对实际应用来说，选择这些参数并不是一件容易的任务，因为目前没有权威性的办法可用。一方面，对优化软件包来说，重要的是要搜索寻优空间足够大的一部分，这预示着 NC 和 CNI 值都要大；另一方面，n 必须选得足够大，以便对于每一个特定的配置 v_1，v_2，…，v_k，能够得到 $E[R(v_1, v_2, \cdots, v_k)]$ 足够精确的估计。否则，优化算法可能会在搜索空间中以一个完全不合适的方向的推进(对于这个问题，建议先做一些初始实验，以确定一个合适的 n 值)。如果人们只能做固定数目的模型重复运行就试图找到一个接近最优解，那么这两个目标(搜索许多配置与每个配置做许多的重复运行)可能相互冲突。

对某些应用来说，当前采用基于仿真的优化的另一个困难是每次重复运行的执行时间。例如，对军事系统和通信网络的某些仿真模型来说，很长的执行时间并不难见。然而，随着计算机速度持续变快，这个困难应该变得不那么严重了。另一方面，它又会部分地被人们对更大更复杂的系统建模所抵消。

尽管基于仿真的优化目前有很多困难，我们仍然相信，当人们需要系统而有效的方法来确定大量的系统配置中哪一个有接近的最优值时，它可以是一个相当有价值的工具。

习题

12.1 回顾习题1.22中的制造系统模型，有5台可能发生故障停机的机器，以及s个维修人员。假设车间还没有建成，除了决定要雇佣多少维修人员外，管理部门要做如下两个决策。

(a) 市场上有一种高质量的“豪华”机器，更加可靠，它的运行时间为均值为16小时的指数分布(而不是标准机器的8小时)。然而，这些豪华机器价格较高，这意味着每台豪华机器故障停机每小时花费$100(而不是$50)。由于豪华机器工作并不快，车间仍然还需要5台。还假设车间不能够对每种机器都购买一些，即机器必须要么都是标准的，要么都是豪华的。

(b) 除了雇佣标准的维修人员外，经理还有一个选择，即雇佣一组更加训练有素的“专家级”维修人员，必须付他们每小时$15(而不是标准维修人员的每小时$10)，但是他们维修故障机器的时间可为均值为1.5小时的指数分布(而不是2小时)。维修人员必须要么全部为标准的，要么全部为专家。

使用表12.28中的编码来执行全2^3析因实验，重复$n=5$次，并计算所有期望的主效用和交互效用的90%置信区间。每个仿真运行800小时，5个机器按照工作顺序开始运行。保证所有运行是独立的，你的结论是什么？

表12.28 通用机器维修模型的编码表

因子	−	+
s	2	4
机器类型	标准	豪华
维修人员类型	标准	专家

12.2 对于第2.5节的分时计算机模型，假定正在考虑调整服务量长度q(如在习题10.3中那样)以及采用在习题2.18和习题10.5中所讨论的另一种处理策略。执行该两因子2^2析因设计($q=0.005$或者0.40，处理策略要么是原来在第2.5节中所描述的策略；要么是习题2.18中的策略)，运行模型完成500作业(无预热)，使用35个终端，所有终端初始都在思考状态。每个设计点进行$n=5$次重复运行，并构造期望主效用和交互效用90%置信区间；在设计点的仿真应该是独立的。

12.3 对于例12.2和例12.3的库存模型，考虑整个2^2析因设计的四个设计点都采用公共随机数(CRN)。令$C_{12}=\mathrm{cov}(R_1, R_2)$，$C_{13}=\mathrm{cov}(R_1, R_3)$，等等，假设CRN“工作”，即$R_i$之间的所有协方差均为正。

(a) 根据R_i的上述协方差和方差求主效用以及交互效用两者的方差表达式。关于CRN减小期望效用的估计的方差，你能够做出什么结论？

(b) 假设我们感兴趣的主要是得到期望主效用的精确估计，且不怎么关心期望交互效用。请给出这样做的另一种随机数分配策略的建议。期望交互效用的估计精度会发生什么变化呢？

12.4 考虑例9.26的通信网络，其中每个SP(信号点)有两个处理器。为了最大程度的减小平均端到端延迟，第三个处理器应当安装在哪个SP上呢？执行2^4析因设计，在16个设计点每个点进行$n=10$次重复运行；应该使用CRN仿真不同的系统配置(参见例11.9)。所有的重复运行长度都是62秒，前2秒为预热期。构造期望主效用和两因子交互效用的90%置信区间。哪些效用是统计显著的？

12.5 考虑有5个单服务台站的串行排队系统，每个都有自己的FIFO队列。假设到达系统(站1)的时间间隔为均值为10分钟的指数分布。进一步假设所有的服务时间都是指数分布的，站1至站5的平均服务时间分别为8、6、9、7、5分钟。系统初始为空且闲，运行正好100 000分钟。

(a) 为了最大限度地减小在系统中的平均时间，第二个服务台应当加在哪个站上？进行2_{V}^{5-1}部分因子设计，在16个设计点的每个点都进行$n=5$次重复运行；应该使用CRN仿真不同的系统

配置。构建期望主效用和两因子交互效用的90%置信区间。哪个效用是统计显著的？

(b) 原来的系统每站一个服务台，计算每个站利用率因子(参见附录1B)。这5个值对你的(a)部分的结果有什么启示？

12.6　考虑例12.10的加工线，其中工作站1到工作站4的机器最优数目分别为3、3、2、2。

(a) 你认为为什么在站2有3台机器比2台机器好呢？

(b) 已知站2是3台机器，为什么在站3不是3台机器呢？

(c) 已知站2应该是3台机器，为什么站1不是2台机器呢？(如果站1和站2分别是2台和3台机器，那么这两个站都会有相同的每小时6个的潜在的加工速率)

12.7　再次考虑例12.10中的加工线，其中 v_5 和 v_6 的最优值分别为7和9(使用枚举法)。

(a) 想一下为什么 v_5 “相对较大”(即更接近10，而不是1)？

(b) 想一下为什么 v_6 “相对较大”？

12.8　我们构造了例12.4中21个期望效用的90%置信区间，所有均基于同一组的 $n=5$ 重复运行。计算不会包括其各自期望效用的置信区间的数目。

12.9　根据第一原理推导出式(12.4)。

12.10　针对库存模型，式(12.10)给出的二阶模型中 $R^2_{\text{adjusted}}=0.864$，如图12.8所示，该二阶元模型似乎为“真”响应面提供了一个较好的表达。另一方面，式(12.7)中给出的元模型中 $R^2_{\text{调整}}=0.923$，但是却为“真”响应面提供了一个较差的表达。如何能够给出对于 R^2 统计的广泛依赖？

12.11　考虑例12.2和例12.3中库存模型的一种推广形式，使其新增两个因子。其中一个为库存估计时间间隔 m，为介于连续库存水平估计间的月份数，用来决定一份订单是否将要被代替。在初始模型中 $m=1$，但是当前考虑的问题是将 m 的值变为2，即仅在每个“其他”月份的初始时间进行估计。第二个新增因子在供应商采用了“快速”交付选项后开始工作。初始条件下，如果有 Z 个商品被订购，则订购成本为 $32+3Z$，交货延迟服从0.5～1个月之间的均匀分布。选择快速交付后，供应商会将交付时间减半(服从0.25～0.5个月之间的均匀分布)，但是收取的费用将相应地增加至 $48+4Z$。因此，交付的优先级 P 为一质量因子，可设为“正常”或“快速”。则在该广义模型中有 $k=4$ 个因子，其水平如下编码如表12.29所示：

表　12.29

因子	2	1
s	20	60
d	10	50
m	1	2
P	正常	快速

对 2^4 的析因设计进行 $n=10$ 次重复运行，并构造期望主效用和交互效用95%的置信区间。改变库存估计时间间隔 m 是否会对平均花费带来较大影响(提示：考虑双向交互效用)？是否有必要使用快速交付选项？

第13章 基于 Agent 的仿真及系统动力学

13.1 引言

这一章将讨论不同于传统离散事件仿真(DES)的其他类型仿真。重点是动态仿真模型，顾名思义，该系统会随着时间变化而不断演变。目前有两大类动态仿真模型，即离散事件仿真(已经在第1章节到第12章节中讨论过了)和连续仿真。回顾离散事件仿真，是指在某些时间点上，其状态变量会发生离散变化。这些时间点是指某一事件(通常情况下)的发生将立即改变系统状态(例如，事件的到达或者离开)的点。然而，我们也可以使用没有真正改变系统状态的虚拟“事件”来调度一个库存估计(参见第1.5节)以及在下午5点钟关闭某家银行的大门(参见第2.6节)。离散事件中仿真钟的推进方式有两种：下一事件推进法(NETA)和固定步长法(FITA)，这两种方法中，以第一种方法使用居多(参见附录1A中对于固定步长法的讨论)。同样，当使用下一事件推进法时，通过调度虚拟事件触发每一个 Δt 时间单元，则可以实现固定步长法(参见第13.2.2小节对于时间推进机制的后续讨论)。

接下来的文章内容安排如下：在第13.2节中我们将讨论离散事件仿真的一个变体——基于 Agent(智能体)仿真。在第13.3节中，我们将讨论连续仿真，它是一种动态仿真模型，其状态变量随着时间变化而连续变化。其中，在第13.3.1小节中，将介绍连续仿真中一种重要的特殊类型——系统动力学。在第13.4节中，将讨论结合离散事件仿真与连续仿真要素的混合仿真模型。在第13.5节中我们将讨论利用随机数来解决某些问题的蒙特卡罗模型。该章节中将会给出两个实例，其一，是蒙特卡罗模型应用于一个可忽略“时间”属性的特定问题中(例如，静态问题)；其二，是蒙特卡罗模型应用于估计随机动态系统的特性。最终，将在第13.6节中讨论电子表格仿真。

13.2 基于 Agent 的仿真

(至少)回溯到2005年，当一部教材和一篇专题追踪论文在冬季仿真会议上发表时，基于 Agent 仿真已经在离散事件仿真界中引起了相当广泛的兴趣(见 www.wintersim.org)，然而，对于基于 Agent 仿真的一般兴趣则可以追溯到更远。出于在会议中对基于 Agent 仿真的这种热情，我们阅读了大量相关文献中的论文和书籍来学习有关这个问题的更多知识。但是，我们非常惊奇地发现目前没有一个标准有可接受的有关 agent 或者基于 Agent 仿真的定义。另外，有大量的表达方式，人们经常联想到基于 Agent 仿真(例如，复杂适应系统和涌现)，当时根本不清楚哪些表达方式是基于 Agent 仿真真正的核心所在。后来，极少有文献写到如何在一个基于 Agent 仿真中推进时间，具体到，对于下一事件，推进方法在一般情况下是否适用。由于缺乏决定性信息，我们随后采访了25位之多的基于 Agent 仿真的实践者及基于 Agent 仿真或者离散事件仿真的软件研发人员。在下面的讨论中，我们提供了一份通过阅读以及采访所总结出的材料大纲(我们想要感谢美国阿贡国家实验室的 Charles Macal 博士和 Michael North 博士，以及 InfoLogix 的 Douglas Samuelson 博士在准备这一章节的材料过程中，对我们给予的热情帮助)。

智能体是一个具有自主性的“实体”，它可以感知周围的环境，包括其他的智能体，并根据感知到的信息作出决策。智能体拥有若干属性，以及一系列决定自身行为的基本

if/then 规则。它们也可以随着时间的推移而学习(更好的理解其他智能体的状态和周围环境)和采用自身的行为(改变自身决策规则)，这就要求它们需具有某种记忆形式。可能的智能体的例子包括人、动物、车辆及组织。我们将智能体仿真定义为一种离散事件仿真，实际上，其实体(智能体)以主要方式与其他实体及其周围环境进行交互。我们称基于Agent仿真是离散事件仿真的一个变体，因为世界上存在的所有基于Agent仿真，其状态着实是在某些时间点上发生改变。Mcal，North与Samuelson在2013年发表的文章中，以及Beeker，Page在2006年发表的文章中都支持了我们对于基于Agent仿真与离散事件仿真之间的关系的观点。同样，Simio和Arena仿真软件的发明者Dennis Pegden博士于2010年在其发表的文章中称："基于Agent仿真只是离散事件仿真的一个特例。"

下面的例子说明某些离散事件仿真的实体具备自主性并与其他智能体及其周围环境进行交互。此外，通过在系统中的移动，这些实体也在不断地学习。

例 13.1 考虑如下假定队列系统。用户(实体)以随机方式到达一包含5个单一服务队列的系统，队列编号分别为从1到5。一位用户需要在10台连续的服务器中任选一台完成一个工作单元。例如，一位用户可能会选择在3号服务器上完成第一个工作单元，在2号服务器上完成第二个工作单元，在3号服务器上完成第三个工作单元，以此类推。唯一的限制条件是不可以在同一台服务器上同时完成两个连续工作单元。当用户全部完成10个工作单元后离开系统。假设用户从一台服务器过渡到另一台服务器的时间为0。

第$i(i=1, 2, \cdots, 5)$台服务器可以在t_i分钟内连续完成一个工作单元。t_i的初始值未知，并假设被用户赋值为2。在一位指定用户使用第i台服务器后，知道了从此之后t_i的实际值。在一位顾客在指定服务器上完成一工作单元后，便可以选择作为预期总处理时间最短的服务器作为下一台工作用服务器(如果有需要的话，在关联过程中，服务器是随机选择的)。为了选择服务器，正在使用另一台服务器的用户假设有半个剩余工作单元需要完成。例如，服务器i上有四项工作需要完成，然后，剩余总处理时间假设为$3.5t_i$分钟，假设t_i的值或为已知或被赋为2。

该系统通常情况下会使用离散事件仿真进行建模。但是，该例中的实体具有自主性，并基于它们对周围环境状态的感知而选择相应的服务器，同时在从一台服务器移动到另一台服务器的过程中不断学习周围环境，有助于它们做出更好的决策。该例可以精炼如下：假设一位用户在第i台服务器上完成了一个工作单元，t_i的实际值便传递给了系统中的其他实体。因而，该系统中的离散事件仿真实体拥有了智能体通常所具备的特性。◀

在基于Agent仿真建模中，我们使用了*自下而上的方法*，该方法的核心是描述各独立智能体的行为和交互性。复杂的基于Agent仿真通常借助于面向对象的软件(例如，Java)，实体*变量*(或数据)对应属性，*方法*对应行为。或许也可以在商用离散事件仿真软件包中实现基于Agent仿真，但是某些时候，也可能会很复杂，过程也可能会变得曲折。一个潜在的问题是绝大多数的离散事件仿真软件包并非是面向对象的。实体确实拥有属性，但是行为并不是封装在实体中，而是在实体通过的模"块"中实施的。相反，机场安保的基于Agent仿真则是成功地在ExtendSim通用仿真软件中实施的[参见第3.5.2小节和Weiss(2011)]。

在某些基于Agent仿真中，随着时间的推移，"低级别"的智能体之间的交互会导致整个系统发生*涌现行为*，该行为是*无法*从独立智能体的特性中推断出来的。有人可能会认为若系统级行为发生涌现，则该行为必须是意想不到的或者与众不同的。但是，某些事情是否是意想不到的很大程度上依赖于观察家的直觉。现在，我们给出基于Agent仿真中最典型的涌现例子之一(参见John Conway的《生命游戏》中"滑翔机"的涌现[Gardner(1970)])。

例 13.2 考虑"群体"鸟群模型的变体[Reynolds(1987)]中的智能体(群体)遵循以下三种简单规则：

内聚力——每个智能体均控制在“本地”(或附近)的平均位置；

分离——每个智能体之间均互相避开，避免出现本地智能体聚集；

对齐——每个智能体均控制在本地智能体的平均航向上。

对于一个指定智能体来说，在某一明确范围内的其他智能体均被认为是本地智能体。甚至于智能体对邻近的其他智能体仅应用这三个简单规则，许多人都认为无领航者群体是一个涌现行为的例子。利用NetLogo软件生成图13.1所示的这种合群性(2013)。◀

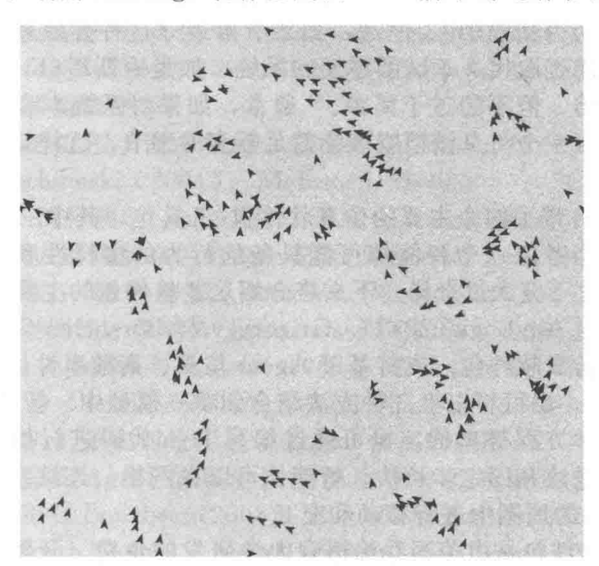


图13.1 群体模型生成的无领航者群体

针对为被仿真的系统提供有用知识的这一问题，我们认为基于Agent仿真没有必要表现出与众不同的行为。此外，一般的离散事件仿真也同样可以表现涌现行为，比如意外的瓶颈、死锁以及振荡。

在下述几种情况下可能会用到基于Agent仿真：

- 当“系统”中的实体可以与其他实体及其周围环境进行交互时；
- 当实体有必要学习和采用自身行为时；
- 当实体的移动依赖于态势感知(对周围环境的感知)而非“脚本”时。

最后一种情况在军事应用中尤为重要。例如，陆地战的仿真模型。历来，在许多模型中，士兵的移动绝大多数情况下都是依赖于脚本的。例如，一名士兵并不是按照战术部署在移动，而是可能在其“标记”位置开始，首先移动到预定路径上的第一个点，再移动到预定路径上的第二个点，以此类推进行移动。另一方面，在基于Agent仿真中，一个智能体(士兵)的路线移动和战术行动可以依赖于智能体的“个性”特征(例如，一个智能体倾向于接近友军智能体并避开敌军智能体)。此外，这些特征在回应一个仿真事件时而发生改变，比如某智能体被枪击时。

我们现在所讨论的是某些与基于Agent仿真或者智能体密切相关的附加项的重要性，这些附加项包括复杂自适应系统、适应性，以及基于本地信息的简单规则(上文中所讨论的涌现是基于Agent仿真中的一个常用术语)。基于Agent仿真领域曾经受到了复杂自适应系统学科的极大影响。复杂自适应系统(CAS)是具有交互性智能体的系统，这些智能体的集体行为表现在适应性、自我组织性，以及涌现性。如果某事物自身变得越来越有组织性，则称其具有自我组织性。现实世界中复杂自适应系统的例子包括证券交易所，社会性昆虫和蚂蚁，生态系统，制造业以及大脑。自此，基于Agent仿真已广泛用于研究现实世界中的复杂自适应系统，尤其是人工生命[参见Holland(1995)和Macal(2009)]，有些人

一直认为基于 Agent 仿真应该主要应用于此类系统的建模。但是，我们并不认同这种说法。例如，有许多重要的书籍[例如，Epstein 和 Axtell(1996)，以及 Gilbert 和 Troitzsch(2005)]和论文鲜少提到复杂自适应系统。同样，仅仅因为复杂自适应系统表现出适应行为并不一定意味着所有的智能体都必须如此[例如，参见 Macal 和 North(2010)中称：智能体可以具有适应行为]。另外，也有些文章称，智能体应使用仅基于“本地”信息的简单规则，比如例 13.2 中的群体模型[参见 Epstein 和 Axtell(1996，24 页)所述的模型中，其智能体的合理性是人为受限的]。但是，如果严格遵守这种建模理念，那么如何得知这种仿真模型得出结果能否取代真正试图建模的系统？如爱因斯坦(Einstein)所说的：“任何事情都应尽可能的简洁，但不能过于简单。”通常，如果对系统本身进行的测试是可行的且有成本效益的，那么一个仿真模型应该涵盖足够多的细节，这样该模型得出的结论才能够接近实际需求。

基于 Agent 仿真自带了四个主要的仿真软件包/工具包，其中三个需在 Java 的面向对象编程环境中运行。当考虑一个智能体可能具备的行为的多样性和复杂性时，构建基于 Agent 仿真的编程方法不应太过奇异。下文将介绍这些软件包的主要特征。

AnyLogic 软件包[AnyLogic(2013)，Grigoryev 与 Borshchev(2013)]是由 AnyLogic 公司发行的一种商业仿真软件包，支持基于 Agent 仿真、离散事件仿真，以及系统动力学(参见第 13.3.1 小节)，也可将以上三种方法结合到单一模型中。智能体行为是由状态表、流程图或者系统动力学方程得出的，也可通过编写 Java 代码进行扩展。同时还支持时间管理中的下一事件推进法和固定步长法。智能体可以在网格、连续空间、网络，以及地理信息系统(GIS)显示出的地图中进行移动和交互。

MASON(2013)软件包是由美国乔治梅森大学研发的免费、开源工具包，可提供构建基于 Agent 仿真中的核心服务。该软件使用下一事件推进法进行时间推进，并提供二维和三维空间可视化工具的可选套件。MASON 模型可构建在网格、连续空间、网络以及地理信息系统上。MASON 软件包的用户手册多达 300 页，其中在第 8 页中写道：“MASON 的用户需具有良好的 Java 知识基础。”

NetLogo(2013)软件包是由美国西北大学的 Uri Wilensky 教授于 1999 年研发的免费、开源软件包，可构建基于 Agent 仿真和基础系统动力学模型。这些模型由已被扩展为支持智能体的 Logo 编程语言的一种方言构建而成。该软件简单易学，包含优秀的文档以及带有大量仿真模型实例的模型库。NetLogo 模型具有一个界面，包含按钮、开关、滑动条、计算器、绘图器，以及动画制作工具。该模型可构建在网格、连续空间、网络以及地理信息系统上，但仅支持固定步长法进行时间推进。

对于大量的、不同的模型输入参数设置，NetLogo 软件包的“行为空间”工具利用多重处理器可以自动将其逐一复制成多个副本上(对于例 12.5 中的捕食开发模型以及第 13.2.1 小节，输入参数为羊群能量增益、羊群繁殖率、狼群能量增益、狼群繁殖率、青草再生时间)。此外，Railsback 与 Grimm(2012)也谈到了 NetLogo 模型。

Repast Simphony 软件包[Argonne(2013)，North 等(2013)]是由美国阿贡国家实验室和芝加哥大学联合研发的免费、开源软件环境，可构建和分析基于 Agent 仿真。可使用 Java 代码、点和点击流程图、点和点击状态图或者 Logo 编程语言中的 ReLogo 方言进行建模。同时，使用 ReLogo 方言还可以输出 NetLogo 模型。系统动力学模型同样可以通过编程或者 Vensim 软件实现(参见第 13.3.1 小节)。所有这些选项都可以无缝集成到一个单一模型中并在 Java 独立平台中进行编译。

Repast Simphony Java 软件包适用于希望最大程度控制模型细节的建模者，而 ReLogo 软件包适用于希望快速构建仿真模型而不需要学习 Java 细节的建模者。状态图和流程图为建模者提供了可以定义智能体行为的图形工具。标准 Java 语言可以嵌套在 ReLogo 代码、状态图或者流程图中，但是一般不作要求。

Repast Simphony 软件包同时支持下一事件推进法和固定步长法，并且模型可以构建在网格、连续空间、网络以及地理信息系统(GIS)显示出的地图中。而智能体的行为则用包括神经网络系统和遗传算法在内的各种各样的程序库来表示。Repast Simphony 软件包还具备一个带有图形用户接口的参数扫描分析设置系统(类似于 NetLogo 软件包的“行为空间”工具包)。Repast Simphony 软件包还包括一套模型实例以及相应的各种教程、使用手册和其他文档。

Repast Simphony 模型(包括使用 ReLogo 代码编译的模型在内)与 Repast for High Performance Computing[Collier 与 North(2013)]共享同一个核心体系结构，是一种可以在超级计算机上执行的模型程序库。

基于 Agent 仿真曾应用于许多领域，包括防御[Brown 等(2004)；Cioppa 等(2004)；Hill 等(2004)；Ilachineski(2004)；McIntosh 等(2007)；SEAS(2013)]，经济学[Economist(2010)]，流行病学[Bagni 等(2002)；Epstein(2009)]，游戏{例如，星际城市[Lew(1989)]}，包括人群疏散在内的国土安全[Beeker 与 Page(2006)；Carley 等(2006)；Samuelson(2007a，b)]，现代物流[Bonabeau 与 Meyer(2001)]，制造业[Nilsson 与 Darley(2006)]，电影{例如，指环王[Economist(2009)]}，运输[Bonabeau(2002)]，社会科学[Epstein(2007)，Epstein 与 Axtell(1996)]，以及供应链[Siebel 与 Kellam(2003)]。此外，Heath，Hill 与 Ciarallo(2009)；Macal 与 North(2010，2011)，以及 Macal，North，Samuelson(2013)中也讨论了基于 Agent 仿真的应用。

表 13.1 列出了基于 Agent 仿真的发展历程中一些重要的里程碑事件。关于基于 Agent 仿真的重要引用有 Bonabeau(2002)，Epstein 与 Axtell(1996)，Gilbert 与 Troitzsch(2005)，Macal 与 North(2010)，Macal，North，Samuelson(2013)，North 与 Macal(2007)，以及维基百科中的一篇名为《基于 Agent 模型》(2013)的文章。

表 13.1　基于 Agent 仿真发展历程中的若干重要里程碑

时间	里程碑
1970	Conway[Gardner(1970)]研发了*生命游戏*——一个可以表达复杂而有趣的涌现模式的细胞自动机，并被认为是基于 Agent 仿真的一个重要前身
1971	Schelling(1970)研发了 Housing-segregation 基于 Agent 仿真，其中钱币代表智能体，周围环境为跳棋盘。Schelling 展示了重大的长期隔离会发生涌现，而不要求同独立智能体的目标一致
1984	美国圣菲研究所成立，它是专门研究复杂自适应系统的机构，也同样在基于 Agent 仿真的发展历程中占据着重要的一席之地
1986	Reynolds(1987)研发了群体鸟群模型，该模型是典型的涌现(参见例 13.2)
1994	美国圣菲研究所研发了首个执行基于 Agent 仿真的编程语言——SWARM[Minars 等(1996)]。它最初被用于复杂自适应系统建模，尤其是人工生命领域[Macal(2009)]
1994	专门为美国空军设计的系统效能分析仿真(SEAS)研发成功，主要用于空间和 C^4ISR 系统建模，至今仍被广泛应用
1996	Epstein 与 Axtell(1996)研发了开创新的糖域基于 Agent 仿真，主要研究在基于网格的“糖”生长环境中，通过智能体之间的交互，如何实现整个人工社会的从无到有
1999	UriWilensky 在美国西北大学研发了 NetLogo 基于 Agent 仿真编程语言
2000	美国芝加哥大学研发了 Repast 工具包，目前，美国阿贡国家实验室正对其进行深度开发
2000	XJ 科技公司(后来更名为 AnyLogic 公司)在俄罗斯圣彼得堡发布了 AnyLogic 多重范式仿真软件包
2009	Parker 与 Epstein 研发了全球规模智能体模型(GSAM)，可以模拟全球性的流行病并且其人口上限可超过 65 亿[参见 Epstein(2009)、Parker 与 Epstein(2011)]

13.2.1 详细示例

第一个例子是关于可由智能体交互得出的复杂非线性行为。考虑一个来自 NetLogo 仿真软件“模型库”的捕食模型(参见 Wilensky(1997)，在 AnyLogic 与 Repast Simphony 软件的模型库中也包含捕食模型的例子)。狼群和羊群在一个由网格构成的生态系统中共存。该网格的单位元为一块面积为 51×51 的草块。初始条件下，所有草块的存活概率以及死亡概率均为 0.5。以下为假设的默认模型。

- 初始条件下，有 100 只羊和 50 只狼随机分布在网格中，每种动物都朝着随机方向前行(一块草块上可存在多只动物)。
- 每只羊初始时拥有大量的能量均匀分布在集合{0，1，…，7}中。
- 每只狼初始时拥有大量的能量均匀分布在集合{0，1，…，39}中。
- 每只动物在一个整数值仿真钟 tick 向前移动一个单元(例如，固定步长法)。但是不一定会到达新的草块。
- 当一只羊吃掉一块绿草时会增加 4 个单元的能量，每走一步会耗费掉 1 个单元的能量。
- 当一只狼吃掉一只羊时会增加 20 个单元的能量，每走一步会耗费掉 1 个单元的能量。
- 当任意一只动物的能量低于 0 时便会死亡。
- 一只羊每移动一次就会增加 0.04 的概率繁殖后代，而繁殖后代会消耗掉一半的能量。
- 一只狼每移动一次就会增加 0.05 的概率繁殖后代，而繁殖后代会消耗掉一半的能量。
- 任意一只动物每向新的前进方向移动一次，都是在其前一次前进方向的基础上增加一个随机值，该值服从范围在[－50，50]度之间的对称三角形分布(参见第 6.2.2 小节)，并将该范围作为其“视角锥”。
- 一只新出生的后代被赋予一个前进方向服从范围在[0，360]度之间的均匀分布，并以正北方向为起点，沿顺时针方向递增。
- 通常，草块中的草从枯萎到再生会消耗掉 30 个时间单元。但是初始条件下，草块的再生时间是独立的，并均匀分布在时间单位集合{0，1，…，29}上。
- 在每一时间步距中，一只狼会随机选择一只与它同处一块草块上的羊(如果有的话)将其吃掉。

图 13.2 显示了在执行 1 000 个时间单位后，羊群、狼群和四分之一草块的数量(我们将草块的数量平均分为 4 部分进行独立仿真，便于将羊群、狼群和草块这三个要素的数量高效地显示在同一纵轴上)。基于此方案得出的生物数量均处于平衡状态。图 13.3 所示的种群规模，显示狼群的平均能量增益为 30，而其他参数值固定不变。注意：关注更为广泛的种群数量波动。图 13.4所示的种群规模，显示草块的平均再生时间为 10，其他参数值设为其默认值(参见显色板)。由于草块的再生时间更短，所以我们希望羊群的数量有所增加，相应地，狼群的数量也应随之增加。但是，请注意：狼群的死亡率非常高，所以草块和

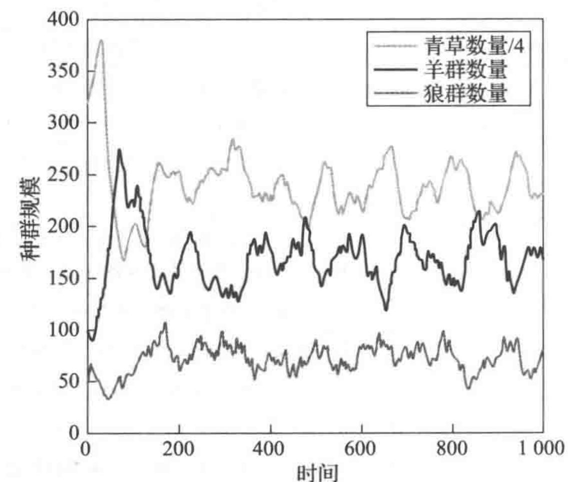


图 13.2 捕食模型中参数值为默认值的条件下，种群规模作为时间函数

羊群的数量达到了新的平衡状态！为什么会出现这种未曾预期到的结果(参见习题 13.1)？

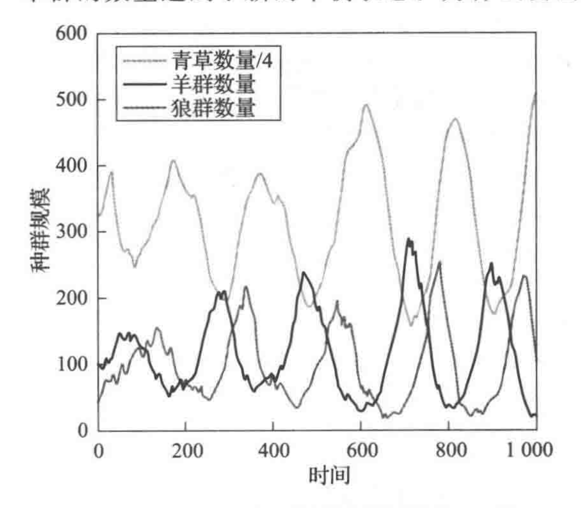


图 13.3 捕食模型中狼群平均增益为 30 的条件下，种群规模作为时间函数

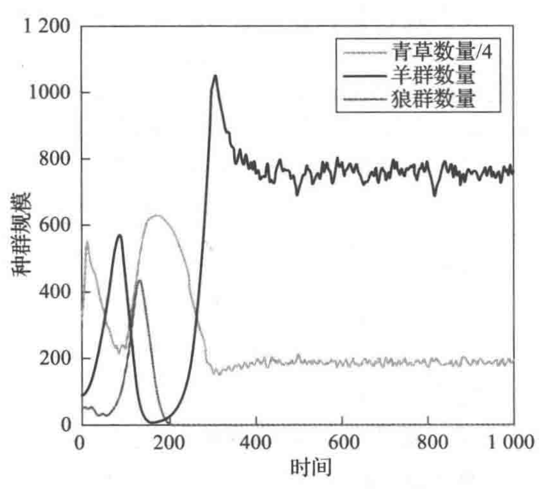


图 13.4 捕食模型中青草再生平均时间为 10 的条件下，种群规模作为时间函数

图 13.5 显示了在第 1 000 个时间单元时，羊群和狼群种群规模的平均值(重复试验次数超过 500)，并将其作为青草再生时间的函数。注意：再生时间大约介于 15～25 之间时的奇异行为(起初，我们设定再生时间的取值范围为 10～50 之间，并将增量设为 2.5，但是后来将平均再生时间的取值范围设定为 18.75～21.25 之间)！

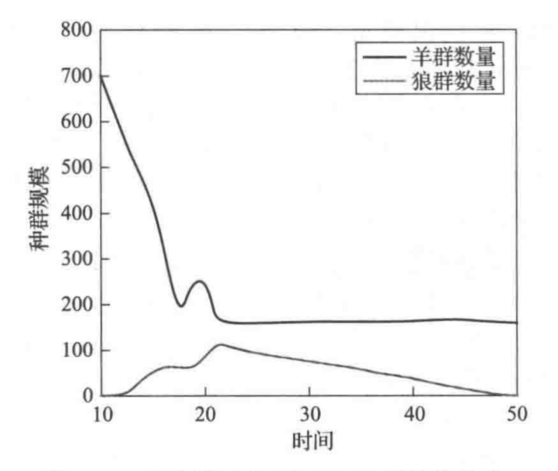


图 13.5 捕食模型中在第 1 000 时间单元时，平均种群规模作为时间函数

给出的第二个例子较为简单，但是它阐明了在仿真运行过程中智能体是如何学习的。考虑 Gilbert 与 Troitzsch[⊖](2005，第 182～190 页)中提到的购物模型。有 10 名顾客和 12 家商铺随机分布在 35×35 网格中，每家商铺只销售一种商品，每名顾客拥有一个包含 10 种想要购买的商品的集合，这些商品是随机生成的并且可重复。我们将讨论该模型的三种变体，这些变体提升了智能体技术的应用水平。

变体 1

如果顾客与销售其所需商品的商铺处于同一网格块中，则顾客可以购买该商品。无论哪种情况，顾客随后朝随机方向前进一个单元。在所有顾客都买到了所需商品后，仿真钟递增一个时间单位(例如，固定步长法)。如果至少有一位顾客还有一件商品需要购买，则所有顾客开始进入另一个购买循环。在所有顾客都买到所需商品后仿真结束，并记录下仿真钟的当前值。

实验结果表明，用不同的随机数对该模型进行 100 余次独立的重复实验(Gilbert 与 Troitzsch 曾进行的实验次数)所需的平均时间为 14，130，这个较大值并不出乎意料，因为每名顾客执行的都是“随机游走”。

⊖ Gilbert N. 与 K. G. Troitzsch. 社会科学家仿真 . 2 版，© 2005，经英国开放大学出版社许可转载。保留所有权利。

变体 2

我们现在为顾客添加感知周围环境的能力。当顾客购买商品时，他首先被置于一个由(x，y)坐标系组成的运行列表中，所有商铺中的商品均分布在周围的 8 个网格块中(这称为它的摩尔(Moore)近邻)。如变体 1 中所述，如果顾客与销售其所需商品的商铺处于同一网格块中，则顾客随后朝随机方向前进一个单元。否则，该顾客便朝向列表中销售其所需商品最近的商铺移动一个单元(如果有的话)。如果顾客列表中所有商铺销售的商品都不是该顾客所需要的，则顾客随后朝随机方向前进一个单元。变体 2 中，所需的平均时间为 6 983，相较于变体 1 下降了 51%。

变体 3

现在允许顾客彼此之间进行通信。尤其是，如果两名顾客处在同一个网格块中，它们互相交换(x，y)坐标，以及相关列表中的商品。变体 3 中，所需的平均时间为 2 013，相较于变体 1 下降了 86%。

13.2.2 基于 Agent 仿真的时间推进机制

如在第 13.1 节中所述，在离散事件仿真中有两种时间推进机制——下一步事件法和固定步长法。几乎所有的基于 Agent 仿真历来使用的都是固定步长法，其原因可能有两个。其一是历史传承，因为(几乎)所有早期的基于 Agent 仿真使用的都是固定步长法，包括一些经典模型，例如，Conway 的生命游戏[Gardner(1970)]，Schelling 的隔离模型[Schelling(1971)]，以及糖域模型[Epstein 与 Axtell(1996)]，而这种传承也一直在延续。其二是有些人觉得在基于 Agent 仿真中使用下一步事件法效率低。也就是说，因为在某些基于 Agent 仿真中，许多智能体之间可能存在交互，就有必要取消和重新排列某些已经置于事件表中并将会在未来的某一时间触发的特定事件。当然，这有可能导致效率降低。下文给出的假设性范例中阐述了在基于 Agent 仿真中使用下一步事件法可能存在的一些困难。

例 13.3 考虑一个由边长为 100 英尺的 50×50 细胞网格组成的假设性环境中，初始条件下，每个细胞变成绿色(图 13.6 和图 13.7 中以白色方框显示)的概率为 0.8，变成红色(以灰色方框显示)的概率为 0.2。绿色细胞变成红色前的时间累积是离散集合{70，71，…，90}分钟上的平均数，即平均值为 80 分钟。红色细胞变成绿色前的时间累积是离散集合{10，11，…，30}分钟上的平均数，即平均值为 20 分钟。

假设有 100 个智能体“存活”在该网格中，并以 60 英尺/分钟的速度移动。初始条件下，智能体随机放置在网格中，并赋予一个均匀分布在离散集合{0°，90°，180°，270°}上的行进方向，以正北方向为起点，沿顺时针方向递增。一个指定的细胞中可存在多个智能体。智能体可以“查找”下一个细胞并确定其颜色。智能体遵守以下移动规则：

- 如果智能体起始于绿色细胞中，则可以朝着确定方向开始移动。
- 如果智能体起始于红色细胞中，则将要等待细胞变为绿色后方可开始移动。
- 如果智能体经过一个细胞后到达其“边缘”时为绿色，那么它只能进入下一个细胞。否则，它将停止并等待，直到下一个细胞变成绿色(指定细胞的右边缘等同于其右边细胞的左边缘)。
- 如果智能体在经过一个细胞的过程中变成红色，它将立即停止移动，直到再次变为绿色后才继续移动。
- 当智能体到达网格的边缘时，系统将计算其在仿真过程的总时间并将其移除。当全部智能体都被移除后，仿真结束，同时将计算出 100 个智能体在“系统”中的平均时间。

假设我们利用下一步事件法对该系统进行仿真。当智能体从一个细胞的“中间位置”开始移动时，我们将它到达当前细胞前向边的时间置于事件列表中(参见图 13.6)。如果智能体到达当前细胞的边缘，并且下一个细胞呈现绿色，则我们将它到达下一个细胞后向边的时间置于事件列表中(参见图 13.7)。如果下一个细胞呈现红色，则智能体停止移动直到

细胞变为绿色为止。如果智能体在经过一个细胞的过程中变成红色，则相应于智能体到达细胞前向边的事件将从事件表中移除。当细胞再次呈现绿色时，智能体到达细胞前向边的事件将重置于事件表中。

对该实例的进一步讨论和数值结果可参见 Macal 与 North(2008)。◀

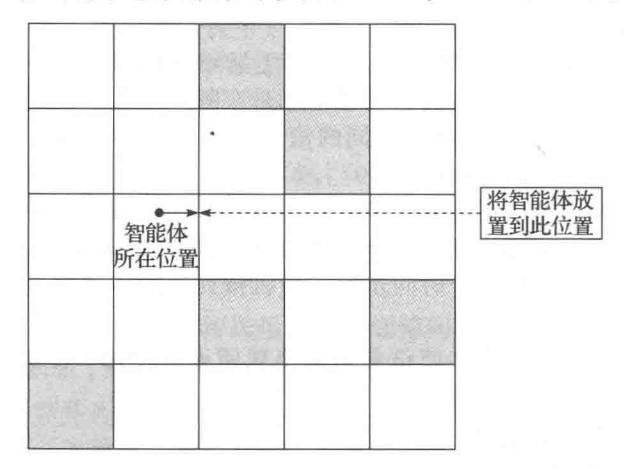


图 13.6　当智能体处在绿色细胞的"中间位置"时的移动

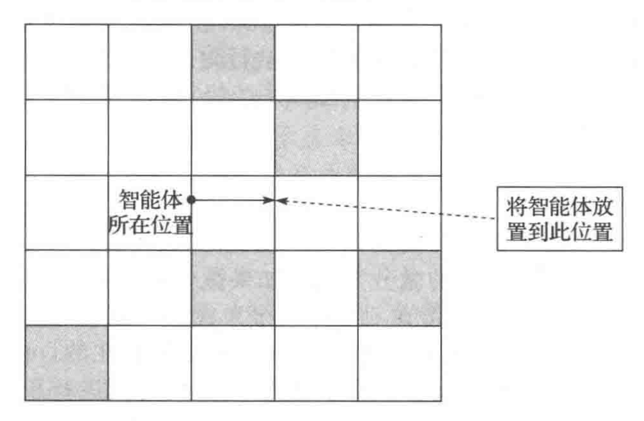


图 13.7　当智能体处在细胞的"边缘"且下一个细胞呈现绿色时的移动

尽管正如在上例中对事件进行需要撤销和重置，共同研发了 Repast Symphony 的 Charles Macal 博士(2013)与 Michael North 博士，以及 AnyLogic 的研发者 Andrei Borshchev 博士(2013)称，一般来说，利用下一步事件法高效执行基于 Agent 仿真是可行的，并建议使用该方法。但是，可能存在某些模型即可以发挥出下一步事件法的优势和也可以发挥出固定步长法的优势。

例 13.4（例 13.3 的延续）现在讨论一些在基于 Agent 仿真中使用固定步长法可能存在的问题。在每一个时间步长中，在检查是否要移动智能体之前，我们都将对是否要更新这些细胞的状态进行检查。例如，假设我们设定 Δt 为 1 分钟，那么考虑更新细胞以及处于每一个时间步长内的智能体的必要条件是什么？◀

细胞

将 Δt 的值设定为 1 分钟确保了每一个细胞都能在合适的时间改变自身状态。但是，我们需要进行大量的检查以确定状态改变的确切时间，由此便导致了计算效率低下。

智能体

如果智能体到细胞前向边的距离大于 60 英尺，则它将向前移动 60 英尺。如果智能体到细胞前向边的距离小于 60 英尺并且下一个细胞呈现绿色，则它将向前移动 60 英尺并处

于另一个细胞中。如果下一个细胞呈现红色，则该智能体将移动到当前细胞的边缘并停止。这将在仿真结果中产生一个错误——由于智能体在1分钟内的移动距离小于60英尺，故导致了其在系统内的停留时间远大于实际需求。当智能体开始移动时，如果其到网格边缘的距离小于60英尺，则该智能体将继续移动到网格边缘，随即销毁。这也将高估智能体在仿真中的运行时间。

假设为了减少不必要的检查(查看是否需要更新细胞的状态)，我们将 Δt 的值设为2分钟。随后，突出的问题是，细胞的状态却不总是在其本应更新的时刻进行更新。通常情况下，在基于Agent仿真中对 Δt 赋予不同的值可能会产生不同的结果，也许不能确定 Δt 的最佳取值[参见Buss与Al Rowaei(2010)]。

在基于Agent仿真中使用固定步长法可能存在的另一个潜在问题是需要在指定的时间点“同时”对许多智能体进行检查，以观察是否它们需要更新，并且其执行顺序并不明显。在这种情况下，也许在每一个时间步距中随机排列智能体的检查顺序会是一个不错的选择。

总之，使用固定步长法可能会使仿真结果出现错误，使用一个较小的 Δt 来改进这一问题将会导致模型的执行速度较慢。

13.2.3 基于Agent仿真的总结

基于Agent仿真的实质是具有自主性的智能体，这些智能体可感知包括其他智能体在内的周围环境，并作出相应的决策。尽管基于Agent仿真似乎是离散事件仿真的一个变体，但这丝毫不会减少基于Agent仿真在研究复杂行为系统过程中的作用。随着时间的推移，低水平智能体之间的交互有时将会导致特殊系统级行为的涌现，但是，基于Agent仿真并不需要生成有用的结果。

13.3 连续仿真

连续仿真关注的是状态变量随时间变化而连续变化的系统建模。连续仿真模型通常涉及描述状态变量变化率与时间关系的微分方程。如果微分方程非常简单，则对其解析求解后可得出在全部时间值下的状态变量值，并将其作为在0时刻状态变量的函数值(参见例13.6)。然而，对于大多数连续仿真模型来说，解析解是不存在的，但是可以利用数值分析技术(例如，龙格-库塔(Runge-Kutta)积分)对数值分析方程进行积分(设置一个小的时间步距)得出在0时刻状态变量的确切值。因此，对连续仿真中的微分方程进行积分，以及离散仿真事件中的固定步长法(参见附录1A)都要用到时间步距。但是，主要区别在于，连续仿真中使用经过计算的方程式对状态变量进行更新，而在离散事件仿真中使用预定事件对状态变量进行更新。

为了建立连续仿真模型，专门设计了若干仿真产品，例如SIMULINK(迈科沃斯软件有限公司)、acslX(神盾科技公司)，以及Dymola(达索系统公司)。除此之外，离散事件仿真软件包Arena[参见Kelton等(2010)]以及ExtendSim[Imagine(2013)]也具备连续建模功能。这两种仿真软件包具有附加功能——允许离散型组建和连续型组建同时存在一单一模型中(参见第13.4节)。对连续仿真应用感兴趣的读者不妨查阅期刊《仿真：国际建模与仿真学会会报》以及《系统动力学综述》。

系统动力学模型

系统动力学模型是一种连续型仿真，用于设计和提升商业领域、政府机构及军事领域的政策或策略(它是一种自上而下的系统建模方法)。相较于离散事件仿真，系统动力学模型从更高的总体层面对系统进行考查，并用来制定更具战略性的决策。大多数的系统动力学模型都是确定性的，但是它们可以拥有随机分量。21世纪50年代，Jay Forrester教授在麻省理工学院的创立了系统动力学模型。商用仿真产品AnyLogic(AnyLogic公司)；iThink(isee系统公司)；Powersim(Powersim软件公司)以及Vensim(Ventana系统公司)

通常用于研发和分析系统动力学模型。更多关于系统动力学模型的信息可以查阅 Sterman 于 2001 年所出的专著或者从国际系统动力学学会(《系统动力学综述》期刊的主办单位)获取。

下文将通过例子来描述系统动力学模型的三个关键要素。

存量是一种“资源”的积累。例子采用的是人口或者动物的数量，产品库存或者储备油的油位(连续变量)。在流图中将存量用一个矩形符号表示(参见图 13.8)。

流量是一种进出存量的资源流。流用带有蝴蝶阀的、粗的、双向箭头(或管道)表示，其中，蝴蝶阀的作用是用来控制通过管道的流速。

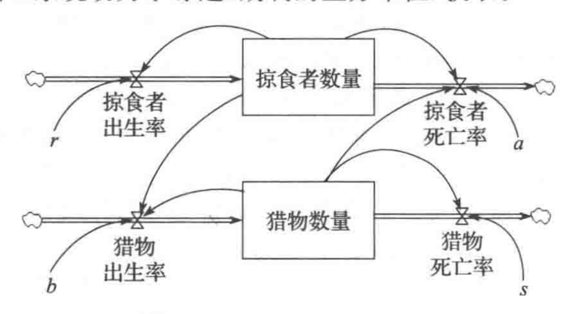


图 13.8 连续捕食模型流图

信息链将存量(或变量)的信息传送给流量的蝴蝶阀，用细的弯箭头表示。

例 13.5 在第 13.2.1 小节中我们考虑了基于 Agent 仿真中捕食系统模型的一种变体。我们现在考虑一种可以称为经典捕食系统的系统动力学模型，该模型是用微分方程来表示的。其周围环境由两个互相交互的种群——捕食者和猎物构成。猎物是被动的，但是它是捕食者的食物来源(例如，捕食者可以是鲨鱼，相应的猎物可以是食用鱼，或者捕食者可以是山猫而其猎物可以是野兔)。设 $x(t)$和 $y(t)$分别为在 t 时刻猎物的数量和捕食者的数量。假设猎物有足够的食物供应，在捕食者数量为 0 的情况下，对于某一确定值 r，它们的繁殖率为 $rx(t)$(我们可以认为 r 的值为自然出生率减去自然死亡率)。由于捕食者和猎物之间的交互，所以就有理由假设由于交互导致的猎物死亡率与两个种群规模的积 $x(t)y(t)$是成正比的。因此，整个猎物种群的变化率 $\mathrm{d}x/\mathrm{d}t$ 记为：

$$\frac{\mathrm{d}x}{\mathrm{d}t}=rx(t)-ax(t)y(t) \tag{13.1}$$

其中，a 为表示比例的正数。

由于捕食者的生存依赖于猎物，所以在猎物数量为 0 的情况下，对于某一确定值 s，它们的繁殖率为$-sy(t)$。另外，两个种群的交互导致了捕食者的数量以一定的比率递增，该比率同样与 $x(t)y(t)$成正比。因此，整个捕食者数量的变化率 $\mathrm{d}y/\mathrm{d}t$ 记为：

$$\frac{\mathrm{d}y}{\mathrm{d}t}=-sy(t)+bx(t)y(t) \tag{13.2}$$

其中，b 为大于 0 的数。

式(13.1)和式(13.2)即为著名的洛特卡-沃尔泰拉(Lotka-Volterra)方程[参见 Lotka(1925)与 Volterra(1931)]。

初始条件为 $x(t)>0$，$y(t)>0$，式(13.1)和式(13.2)给出了模型的求解表达式。有趣的是，方程中对于所有的 $t\geqslant0$，均有 $x(t)>0$ 及 $y(t)>0$[参见 Braun(1975)]。因此，猎物的数量永远不会消失殆尽。$\{x(t), y(t)\}$的解同样是一个时间的周期函数。即对于所有正整数 n，都存在 $T>0$ 使得 $x(t+nT)=x(t)$，$y(t+nT)=y(t)$。其结果是意料之中的。当捕食者的数量增加时，猎物的数量减少。随之，捕食者的增长率下降，最终造成了捕食者数量的下降。反之，使得猎物的数量有所增加，以此类推。

考虑特定值 $r=1$，$a=0.035$，$s=0.017$，$b=1$，初始种群规模 $x(0)=100$，$y(0)=15$。针对该捕食问题，我们现在以 Vensim 软件为平台研发一个系统动力学模型[参见 Ventana(2013)]。在图 13.8 中我们给出了流图，其中，$x(t)$和 $y(t)$分别表示“猎物种群数量”和“捕食者种群数量”的存量(在 Vensim 软件中称为“盒变量”)。猎物种群的出生率和死亡率分别由“猎物出生数量”和“猎物死亡数量”流图(在 Vensim 软件中称为“比率”)表示，而捕食者种群的出生率和死亡率分别由“捕食者出生数量”和“捕食者死亡数

量”流图表示。例如，如果双击猎物死亡数量流图，将会立即显示出如下方程：

猎物死亡数量$=a\times$猎物种群数量$\times$捕食者种群数量

该信息链(在 Vensim 软件中称为“箭头”)在流图中以弯箭头进行标记，常数 r，a，s，b 在 Vensim 软件中同样以“变量”表示。如果双击变量 r，则自然会显示 1。

我们设定仿真的运行时间为 100 年，在图 13.9 中将 $x(t)$ 和 $y(t)$ 作为时间 t 的函数，其中，$\{x(t), y(t)\}$ 明显呈现出周期性(同样参见彩色插图)。同样可见的是，$y(t)$ 的增速滞后于 $x(t)$ 的增速。图 13.10 显示了带有指定参数的捕食模型的相位图，即 $t\geqslant0$ 时的 $\{x(t), y(t)\}$。图中可见，由于解的周期性，所有的点都落在了“卵形”闭合曲线上。◀

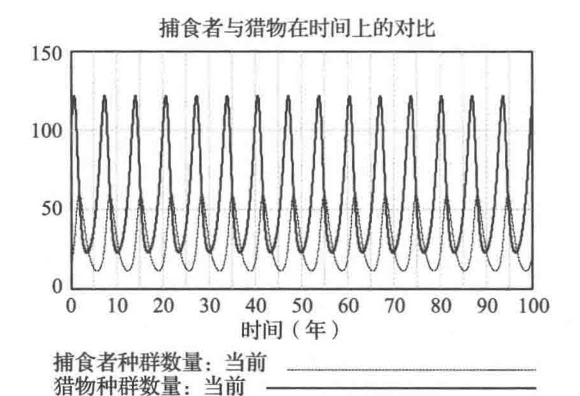


图 13.9　以种群规模作为时间函数的连续捕食模型

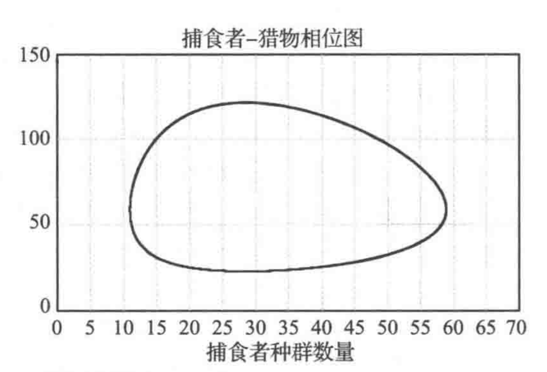


图 13.10　连续捕食模型相位图

例 13.6　我们现在考虑一个瞄准射击作战模型，该模型由 Frederick Lanchester 于 1916 年研发，并在第一次世界大战时将其应用于空战中[参见 MacKay(2006)和 Artelli 与 Deckro (2008)]。**瞄准射击作战**模型描述了一个狙击手直接对敌射击的仿真模型。如果敌人被消灭，狙击手将对新目标进行射击。在目标被清除后，狙击手的火力将越来越多地集中在剩余目标上。锁定目标射击可能应用在双筒步枪、坦克作战，以及空对空作战中。无锁定目标射击是相对于瞄准射击而言的，是狙击手向某一区域射击，可能应用在炮兵火力中。

设 $b(t)$ 和 $r(t)$ 分别为在 t 时刻下，蓝军参战单位的个数和红军参战单位的个数。假设在 t 时刻蓝军部队的损耗率与 $r(t)$ 成正比。因此，将蓝军的兵力变化率 $\mathrm{d}b/\mathrm{d}t$ 定义为：

$$\text{当 } b(t)>0 \text{ 时，}\quad \frac{\mathrm{d}b}{\mathrm{d}t}=-\alpha r(t) \tag{13.3}$$

其中，$\alpha>0$ 为**红军的作战效能**。

同样，将红军的兵力变化率 $\mathrm{d}b/\mathrm{d}t$ 定义为：

$$\text{当 } r(t)>0 \text{ 时，}\quad \frac{dr}{\mathrm{d}t}=-\beta b(t) \tag{13.4}$$

其中，$\beta>0$ 为**蓝军的作战效能**。

当然，$b(t)$ 和 $r(t)$ 必须是非负的。式(13.3)和式(13.4)即为兰彻斯特(战斗动态瞄准目标)方程。

考虑特定值 $\alpha=0.1$，$\beta=0.4$，初始兵力 $b(0)=175$，$r(0)=300$。哪一方能够赢得这场战争？为了得出答案，利用方程(13.3)和(13.4)我们建立了兰彻斯特(Lanchester)方程的 Vensim 软件模型，其流图如图 13.11 所示。在“蓝军死亡率”流图内的方程是有效的。

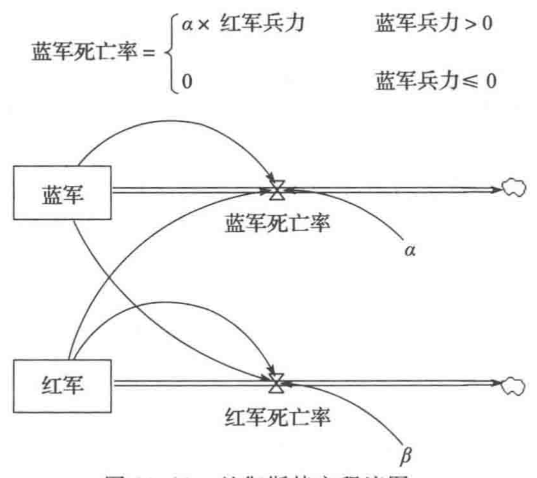


图 13.11　兰彻斯特方程流图

我们将仿真运行时间设置为 12 小时，

图 13.12给出的是关于 t 的函数 $b(t)$ 和 $r(t)$，由图可见，在这场战争中蓝军获胜。然而，当我们将 $b(0)$ 设置为 125 时，其结果如图 13.13 所示：红军赢得了这场战争的胜利。

也许有人会问是否存在一与 $b(0)$，$t(0)$，α，β 有关的分析结果，也就是说，不需要运行该仿真便可分析出这场战争的赢家。经证明[参见 Taylor(1983)]如果 $b(0)/r(0)>\sqrt{\alpha/\beta}$，则在 t_{end} 时刻蓝军将获得胜利。其中，t_{end} 的表达式为：

$$t_{end}=\frac{1}{2\sqrt{\alpha\beta}}\ln\left(\frac{1+\frac{r(0)}{b(0)}\sqrt{\frac{\alpha}{\beta}}}{1-\frac{r(0)}{b(0)}\sqrt{\frac{\alpha}{\beta}}}\right)$$

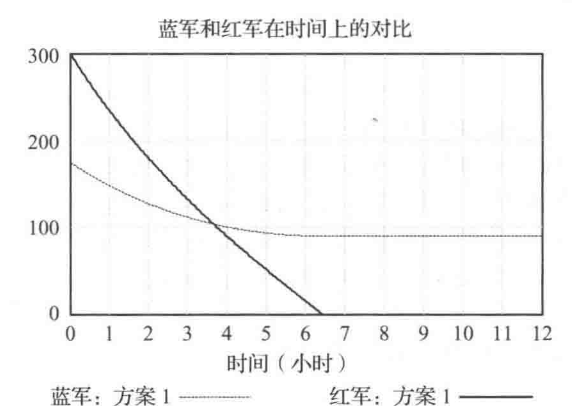


图 13.12　$b(0)=175$，部队规模为时间函数的兰彻斯特方程

蓝军和红军在时间上的对比
时间（小时）
蓝军：方案 2
红军：方案 2

图 13.13　$b(0)=125$，部队规模为时间函数的兰彻斯特方程

最终的兵力规模为：

$$b(t_{end})=\sqrt{b^2(0)-\frac{\alpha}{\beta}r^2(0)},\quad r(t_{end})=0$$

因此，如图 13.12 所示：如果取 $b(0)=175$，$r(0)=300$，$\alpha=0.1$，$\beta=0.4$，则，$b(0)$ $b(0)/r(0)=0.58>0.50=\sqrt{\alpha/\beta}$，$t_{end}=6.41$ 小时，$b(t_{end})=90.14$，$r(t_{end})=0$。反之，如果 $b(0)/r(0)=\sqrt{\alpha/\beta}$，则红军将赢得胜利。最后，如果 $b(0)/r(0)=\sqrt{\alpha/\beta}$，则当 $t\to+\infty$ 时，$b(t)$ 和 $r(t)$ 的值都为 0。◀

13.4　离散-连续混合仿真

因为某些系统既不是完全离散的也不是完全连续的，所以就有必要构建一个同时满足离散事件和连续事件的仿真，离散-连续混合仿真由此得来。Pritsker(1995，61～62 页)描述了三种基本的交互类型，这些类型可同时适用于离散变化和连续变化的状态变量：

- 一个离散事件可能导致在连续状态变量值上发生离散变化。
- 一个离散事件可能导致在某一特定时间，其关联性支配一个连续状态变量的变化。
- 连续状态变量达到阈值时可能导致一个离散事件的发生或预发生。

离散-连续混合仿真可在 Arena 软件[Kelton 等(2010)]和 ExtendSim 软件[Imagine (2013)]中实现建模。

下文给出的离散-连续混合仿真示例是对一模型进行简单描述，该模型是由 Pritsker (1996，354～364 页)给出的，除此模型外，Pritsker 还给出了有关此类模型的其他示例。

例 13.7　运输原油的邮轮到达一单一港口，提供一个经过管道流入炼油厂的储油罐。一个正在卸油的邮轮以特定的恒定速率将油运输到储油罐中(当港口忙于形成队列时而到达的邮轮)。储油罐以不同的特定速率为炼油厂提供原油。港口的开放时间为早 6 点到午

夜12点，出于安全考虑，当港口关闭后邮轮停止卸油。

该(简易)模型的离散事件为一个待卸油的邮轮的到达、在午夜关闭港口、在早6点开放港口，将待卸油邮轮以及储油罐内的储油位设置为连续静态变量，其变化率由微分方程来表示[详见 Pritsker(1995，354～364页)]。当邮轮内储油位低于其容量的5%时，视为邮轮卸油完成，但是，如果储油罐内的储油位达到其容量时，卸油必须暂时停止。当邮轮的储油位下降到其容量的80%时，恢复卸油。如果邮轮内的储油位降到5 000桶以下，炼油厂就必须暂时关闭。为了避免炼油厂频繁开放和关闭，邮轮再次装载50 000桶原油时才恢复向炼油厂供油。与原油储油位相关的5种事件(例如，邮轮中原油储油位低于其容量的5%)均被 Pritsker 称为**状态事件**。不同于离散事件，状态事件不在计划内，但是当连续状态变量经过阈值时却会触发状态事件的发生。◀

13.5 蒙特卡罗仿真

我们将蒙特卡罗仿真定义为采用随机数的方案，即在区间(0，1)上服从均匀分布的随机变量。蒙特卡罗仿真用于解决特定的随机问题或者确定性问题。因此，随机离散事件仿真也属于此定义范畴之列。“蒙特卡罗”仿真或方法最初被命名于第二次世界大战期间，当时该方法用于与发明原子弹相关的问题。对于蒙特卡罗仿真更为详细的讨论，请参见 Hammersley 与 Handscomb(1964)；Halton(1970)；Rubinstein(1981，1992)；Morgan(1984)；Fishman(1996，2006)，Rubinstein 等(1998)；Glasserman(2004)以及 Rubinstein 与 Kroese(2008)。**下例阐述了蒙特卡罗仿真在确定性问题中的应用。**

例 13.8 假设我们想计算积分

$$I=\int_a^b g(x)\mathrm{d}x$$

其中，$g(x)$是一个不可解析积分实值函数(实践表明蒙特卡罗仿真可能不适用于计算单一积分，因为针对此问题有更为高效的数值解析技术，而其更适用于解决带有一个病态行为被积函数的多重积分问题)。为了探讨蒙特卡罗仿真如何解决此类问题，我们设 Y 为一个随机变量$(b-a)g(x)$，其中，X 为在$[a，b]$上服从均匀分布的连续随机变量[记为$U(a，b)$]。则，Y 的期望值为：

$$\begin{aligned}E(Y)&=E[(b-a)g(X)]=(b-a)E[g(X)]\\&=(b-a)\int_a^b g(x)f_X(x)\mathrm{d}x=(b-a)\frac{\int_a^b g(x)\mathrm{d}x}{b-a}\\&=I\end{aligned}$$

表 13.2 应用蒙特卡罗仿真计算积分 $\int_0^\pi \sin x\mathrm{d}x=2$，当 n 取不同值时 $\overline{Y}(n)$ 的取值

n	10	20	40	80	160
$\overline{Y}(n)$	2.213	1.951	1.948	1.989	1.993

其中，$f_X(x)=1/(b-a)$是一个随机变量的概率密度函数，该变量在(a，b)上服从均匀分布(参见第6.2.2小节)[例如，对于第三个等式的证明参见 Ross(2003，45页)]。因此，积分的计算问题曾被简化为估计期望值 E(Y)中的一个步骤。特别是，我们应该使用下式计算 $E(Y)=I$：

$$\overline{Y}(n)=\frac{\sum_{i=1}^{n}Y_i}{n}=(b-a)\frac{\sum_{i=1}^{n}g(X_i)}{n}$$

其中，X_1，X_2，…，X_n 为在$[a，b]$上服从独立同分布的随机变量。

$\overline{Y}(n)$的取值如表13.2所示。(将$\overline{Y}(n)$视为一个的三角域的估计是有益的，该三角域

以$(b-a)$长度为底，$I/b-a$($g(x)$在$[a, b]$上的连续平均值)为高)。另外，该式可证明$E[\overline{Y}(n)]=I$，即，$\overline{Y}(n)$是I的无偏差估计，并且$\text{var}[\overline{Y}(n)]=\text{var}(Y)/n$(参见第 4.4 节)。假设$\text{var}(Y)$是有限的，则当 n 足够大时，$\overline{Y}(n)$将任意接近$I$(概率为 1)(参见第 4.6 节)。

为了从数值上阐明上述方法，假设我们要计算积分：

$$I=\int_0^{\pi}\sin x\mathrm{d}x$$

该式可由值为 2 的初等微积分证明得之。表 13.2 显示了对于不同的 n 值，应用蒙特卡罗仿真计算该积分的结果。 ◀

例 13.9 我们现在利用蒙特卡罗仿真来计算一个随机系统的特征值。考虑图 13.14 所示的系统，该系统欲将信号从 A 传送到 B。5 个组件随机发生故障，并且相互独立，即任意一个发生故障不会影响到其他组件的正常工作。设Y为整个系统发生瘫痪的时间，X_i为组件i发生故障的时间，$i=1, 2, \cdots, 5$。

图 13.14 5 组件系统布置图

表 13.3 5 个组件的平均值(以天为单位)

i	β_i
1	10
2	8
3	7
4	5
5	6

我们假设X_i服从平均天数为β_i的指数分布，β_i的取值如表 13.3 所示。可得(参见习题 13.5)：

$$Y=\min(\max\{X_1,\min[X_2,\max(X_3,X_4)]\},X_5) \tag{13.5}$$

回顾第 8.3.2 小节，利用下式：

$$X_i=-\beta_i\ln U$$

我们可以从一个服从平均数为β_i的指数分布中生成一个随机变量X_i，其中，$U\sim U(0, 1)$。因此，Y的分布中，需要 5 个随机数来生成一个随机变量。根据式(13.5)，我们生成$Y_j(j=1, 2, \cdots, 25\,000)$并且满足$\overline{Y}(25\,000)=4.33$天，为故障时间期望的一个无偏差估计$E(Y)$。假设我们将重复计算系统连续运转 7 天的概率，例如，$p_7=P(Y\geqslant 7)$。将指数函数$I_j(7, +\infty)$定义为：

$$I_j(7,+\infty)=\begin{cases}1, & Y_j\geqslant 7\\ 0, & \text{其他}\end{cases}$$

则无偏差估计p_7为：

$$\hat{p}_7=\frac{\sum_{j=1}^{25\,000}I_j(7,+\infty)}{25\,000}=0.19$$

最后，我们在图 13.15 中列出了当j取 25 000 时Y_j的直方图。 ◀

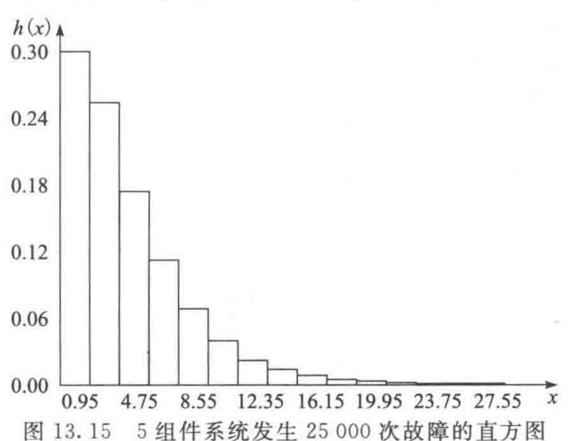


图 13.15 5 组件系统发生 25 000 次故障的直方图

蒙特卡罗仿真广泛应用于解决解析方法学无法解决的统计问题。例如，蒙特卡罗仿真曾用于计算临界值和一个新的假设检验的功效。确定正态性 K-S 检验的临界值(参见第 6.6 节)也是基于蒙特卡罗仿真的。专业读者也可能喜欢阅读科技论文统计年鉴(仿真与计算)，统计计算与仿真学报，以及技术度量学，所有这些期刊中都包含了许多有关蒙特卡罗仿真的示例。

最后，在第 9.4 节中所讨论的方法可以用于确定简单规模 n，使其满足蒙特卡罗仿真研究所需的指定精度。

13.6 电子表格仿真

在目标问题不是太复杂的情况下，可以在 Excel 等电子表格中实现离散事件仿真和蒙特卡罗仿真。关于这一点，Excel 提供了一个随机数生成器，可以从基本的概率分布(例如，正态分布、均匀分布、二项式分布，以及泊松分布)、概要统计(例如，均值与方差)以及图形(例如直方图等)生成随机数。但是，根据 Seila(2005)中所述，电子表格具有如下重要的局限性：

- 只对简单的数据结构适用；
- 难以执行复杂算法；
- 相较于基于离散事件仿真软件包所建立的模型，电子表格仿真运行时间长；
- 数据存储受限。

使用一些辅助软件执行电子表格仿真是非常容易的，例如，知名的电子表格插件@Risk(Palisade 公司)、水晶球软件(Oracle 公司)，以及风险分析软件(一线系统公司)。它们可以提供额外的概率分布，用于对电子表格仿真进行独立复制的简单机制，以及具备分析灵敏度的功能。

电子表格仿真广泛应用于在金融、制造业、工程管理、石油，以及油气等应用领域执行风险分析。更多关于电子表格仿真的信息可以参阅 Evans 与 Olson(2002)、Seila 等(2003)，以及 Winston 与 Albright(2011)。

例 13.10 考虑例 4.23 中的一个简单(s，S)库存系统。如果随机变量 Y 为前 12 个月中每月的平均总费用，则计算如下期望值：

$$E(Y) = \left[\frac{\sum_{i=1}^{12} C_i}{12}\right] = \frac{\sum_{i=1}^{12} E(C_i)}{12}$$

为了计算 $E(Y)$，我们在 Excel 中进行了电子表格仿真，并使用独立随机数进行了 1 000 重复试验。1 000 次试验中，Y 的简单均值和简单方差分别为 99.34 和 28.04[注意：预期的每月总费用的稳态均值为 112.11(参见例 9.3)]。在图 13.16 中，我们给出了使用组宽为 3 对 Y 进行 1 000 次观察后的直方图。对该库存系统的仿真是使用固定步长法的离散事件仿真，Δt 取值为 1 个月。

注意：例 13.9 中 5 组件系统的蒙特卡罗仿真也是使用 Excel 电子表格运行的。◀

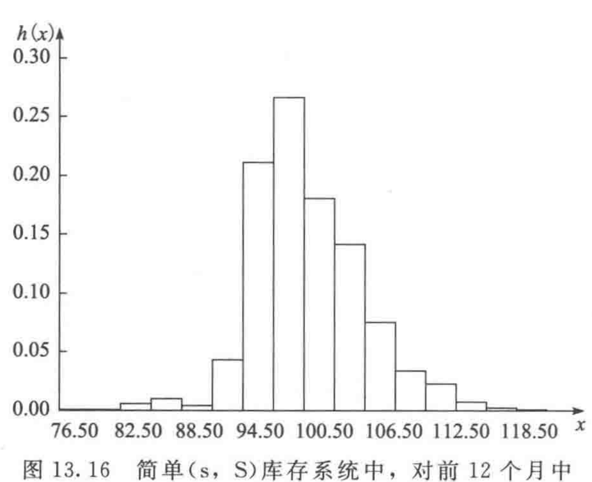


图 13.16 简单(s，S)库存系统中，对前 12 个月中每月平均总费用观察 1 000 次的直方图

习题

13.1　从网站 http://ccl.northwestern.edu/netlogo/下载一个 NetLogo 软件包的免费复制版。在“模型库”中打开模型“狼群-羊群捕食行为”（参见“生物学”)打开“草地?”开关。使用该模型确定为什么当草块的再生时间设置为 10 时，狼群数量快速趋近于 0(参见图 13.4)?

13.2　对于第 13.2.1 小节中的“狼群-羊群捕食行为”，设置狼群数量的初始值为 75，所有其他参数为默认值的条件下重复进行 100 次试验(使用屏幕上方的“工具栏”中的“行为空间”选项)。设 μ_i 为 $i=50$，75 时狼群的最终期望值。利用韦尔奇置信区间对 $\mu_{50}-\mu_{75}$ 构建一个 95%的置信区间(参见第 10.2.2 小节)，能得出什么结论?

13.3　对于例 13.3 中的系统，假设细胞在变为红色之间所呈现出绿色的时间总量服从均值为 80 分钟的指数分布。同样假设细胞在变为绿色之间所呈现出红色的时间总量服从均值为 20 分钟的指数分布。讨论固定步长法中难以选择 Δt 的原因。

13.4　对于例 13.3 中的方法 2[例如，$b(0)=125$]，t_{end} 和 $r(t_{\text{end}})$ 的值分别是多少?

13.5　证明例 13.5 中 Y 的表达式成立。

13.6　对于例 13.9 中的 5 组件系统，下面 3 个选项中，哪个选项将最大限度地增加故障的平均时间：

(a) 用两个并联的组件代替组件 1。

(b) 用两个并联的组件代替组件 2。

(c) 用两个并联的组件代替组件 5。

对于每一个选项，使用 $n=25\,000$ 的蒙特卡罗仿真。如果组件 i 被复制($i=1$，2，5)，则设 μ_i 为故障的平均时间。使用韦尔奇方法和邦费罗尼不等式(参见第 9.7 节)。构建 $\mu_1-\mu_2$，$\mu_1-\mu_5$，$\mu_2-\mu_5$ 的置信区间，则综合置信度至少为 95%。

13.7　执行蒙特卡罗仿真计算常量 π 的值。提示：考虑一个边长为 1 的正方形。在该正方形内画一个半径为 1/2 的圆形。圆形的面积与正方形的面积之比为 $\pi/4$。当 $n=25\,000$ 时计算该值。

附　录　相关分布的临界点

表 A.1　自由度为 ν 的 t 分布的临界点 $t_{\nu,\gamma}$ 和标准正态分布 $\gamma=P(T_\nu\leqslant t_{\nu,\gamma})$ 的临界点 z_γ，T_ν 是自由度为的 ν 的 t 分布的随机变量；最后一行，$\nu=\infty$，给定一个正态分布临界点满足 $\gamma=P(Z\leqslant z_\gamma)$，这里，$Z$ 为一个标准正态分布变量

ν	γ															
	0.600 0	0.700 0	0.800 0	0.900 0	0.933 3	0.950 0	0.960 0	0.966 7	0.975 0	0.980 0	0.983 3	0.987 5	0.990 0	0.991 7	0.993 8	0.995 0
1	0.325	0.727	1.376	3.078	4.702	6.314	7.916	9.524	12.706	15.895	19.043	25.452	31.821	38.342	51.334	63.657
2	0.289	0.617	1.061	1.886	2.456	2.920	3.320	3.679	4.303	4.849	5.334	6.205	6.965	7.665	8.897	9.925
3	0.277	0.584	0.978	1.638	2.045	2.353	2.605	2.823	3.182	3.482	3.738	4.177	4.541	4.864	5.408	5.841
4	0.271	0.569	0.941	1.533	1.879	2.132	2.333	2.502	2.776	2.999	3.184	3.495	3.747	3.966	4.325	4.604
5	0.267	0.559	0.920	1.476	1.790	2.015	2.191	2.337	2.571	2.757	2.910	3.163	3.365	3.538	3.818	4.032
6	0.265	0.553	0.906	1.440	1.735	1.943	2.104	2.237	2.447	2.612	2.748	2.969	3.143	3.291	3.528	3.707
7	0.263	0.549	0.896	1.415	1.698	1.895	2.046	2.170	2.365	2.517	2.640	2.841	2.998	3.130	3.341	3.499
8	0.262	0.546	0.889	1.397	1.670	1.860	2.004	2.122	2.306	2.449	2.565	2.752	2.896	3.018	3.211	3.355
9	0.261	0.543	0.883	1.383	1.650	1.833	1.973	2.086	2.262	2.398	2.508	2.685	2.821	2.936	3.116	3.250
10	0.260	0.542	0.879	1.372	1.634	1.812	1.948	2.058	2.228	2.359	2.465	2.634	2.764	2.872	3.043	3.169
11	0.260	0.540	0.876	1.363	1.621	1.796	1.928	2.036	2.201	2.328	2.430	2.593	2.718	2.822	2.985	3.106
12	0.259	0.539	0.873	1.356	1.610	1.782	1.912	2.017	2.179	2.303	2.402	2.560	2.681	2.782	2.939	3.055
13	0.259	0.538	0.870	1.350	1.601	1.771	1.899	2.002	2.160	2.282	2.379	2.533	2.650	2.748	2.900	3.012
14	0.258	0.537	0.868	1.345	1.593	1.761	1.887	1.989	2.145	2.264	2.359	2.510	2.624	2.720	2.868	2.977
15	0.258	0.536	0.866	1.341	1.587	1.753	1.878	1.978	2.131	2.249	2.342	2.490	2.602	2.696	2.841	2.947
16	0.258	0.535	0.865	1.337	1.581	1.746	1.869	1.968	2.120	2.235	2.327	2.473	2.583	2.675	2.817	2.921
17	0.257	0.534	0.863	1.333	1.576	1.740	1.862	1.960	2.110	2.224	2.315	2.458	2.567	2.657	2.796	2.898
18	0.257	0.534	0.862	1.330	1.572	1.734	1.855	1.953	2.101	2.214	2.303	2.445	2.552	2.641	2.778	2.878
19	0.257	0.533	0.861	1.328	1.568	1.729	1.850	1.946	2.093	2.205	2.293	2.433	2.539	2.627	2.762	2.861
20	0.257	0.533	0.860	1.325	1.564	1.725	1.844	1.940	2.086	2.197	2.285	2.423	2.528	2.614	2.748	2.845
21	0.257	0.532	0.859	1.323	1.561	1.721	1.840	1.935	2.080	2.189	2.277	2.414	2.518	2.603	2.735	2.831
22	0.256	0.532	0.858	1.321	1.558	1.717	1.835	1.930	2.074	2.183	2.269	2.405	2.508	2.593	2.724	2.819
23	0.256	0.532	0.858	1.319	1.556	1.714	1.832	1.926	2.069	2.177	2.263	2.398	2.500	2.584	2.713	2.807
24	0.256	0.531	0.857	1.318	1.553	1.711	1.828	1.922	2.064	2.172	2.257	2.391	2.492	2.575	2.704	2.797
25	0.256	0.531	0.856	1.316	1.551	1.708	1.825	1.918	2.060	2.167	2.251	2.385	2.485	2.568	2.695	2.787
26	0.256	0.531	0.856	1.315	1.549	1.706	1.822	1.915	2.056	2.162	2.246	2.379	2.479	2.561	2.687	2.779
27	0.256	0.531	0.855	1.314	1.547	1.703	1.819	1.912	2.052	2.158	2.242	2.373	2.473	2.554	2.680	2.771
28	0.256	0.530	0.855	1.313	1.546	1.701	1.817	1.909	2.048	2.154	2.237	2.368	2.467	2.548	2.673	2.763
29	0.256	0.530	0.854	1.311	1.544	1.699	1.814	1.906	2.045	2.150	2.233	2.364	2.462	2.543	2.667	2.756
30	0.256	0.530	0.854	1.310	1.543	1.697	1.812	1.904	2.042	2.147	2.230	2.360	2.457	2.537	2.661	2.750
40	0.255	0.529	0.851	1.303	1.532	1.684	1.796	1.886	2.021	2.123	2.203	2.329	2.423	2.501	2.619	2.704
50	0.255	0.528	0.849	1.299	1.526	1.676	1.787	1.875	2.009	2.109	2.188	2.311	2.403	2.479	2.594	2.678
75	0.254	0.527	0.846	1.293	1.517	1.665	1.775	1.861	1.992	2.090	2.167	2.287	2.377	2.450	2.562	2.643
100	0.254	0.526	0.845	1.290	1.513	1.660	1.769	1.855	1.984	2.081	2.157	2.276	2.364	2.436	2.547	2.626
∞	0.253	0.524	0.842	1.282	1.501	1.645	1.751	1.834	1.960	2.054	2.127	2.241	2.326	2.395	2.501	2.576

表 A.2　自由度为 ν 的卡方分布 $\gamma=P(Y_\nu\leqslant X^2_{\nu,\gamma})$ 的临界点 $X^2_{\nu,\gamma}$，Y_ν 为自由度为的 ν 的卡方分布的随机变量；对于较大 ν 的值，用 $X^2_{\nu,\gamma}$ 的近似值，参见 7.4.1 节

ν	γ						
	0.250	**0.500**	**0.750**	**0.900**	**0.950**	**0.975**	**0.990**
1	0.102	0.455	1.323	2.706	3.841	5.024	6.635
2	0.575	1.386	2.773	4.605	5.991	7.378	9.210
3	1.213	2.366	4.108	6.251	7.815	9.348	11.345
4	1.923	3.357	5.385	7.779	9.488	11.143	13.277
5	2.675	4.351	6.626	9.236	11.070	12.833	15.086
6	3.455	5.348	7.841	10.645	12.592	14.449	16.812
7	4.255	6.346	9.037	12.017	14.067	16.013	18.475
8	5.071	7.344	10.219	13.362	15.507	17.535	20.090
9	5.899	8.343	11.389	14.684	16.919	19.023	21.666
10	6.737	9.342	12.549	15.987	18.307	20.483	23.209
11	7.584	10.341	13.701	17.275	19.675	21.920	24.725
12	8.438	11.340	14.845	18.549	21.026	23.337	26.217
13	9.299	12.340	15.984	19.812	22.362	24.736	27.688
14	10.165	13.339	17.117	21.064	23.685	26.119	29.141
15	11.037	14.339	18.245	22.307	24.996	27.488	30.578
16	11.912	15.338	19.369	23.542	26.296	28.845	32.000
17	12.792	16.338	20.489	24.769	27.587	30.191	33.409
18	13.675	17.338	21.605	25.989	28.869	31.526	34.805
19	14.562	18.338	22.718	27.204	30.144	32.852	36.191
20	15.452	19.337	23.828	28.412	31.410	34.170	37.566
21	16.344	20.337	24.935	29.615	32.671	35.479	38.932
22	17.240	21.337	26.039	30.813	33.924	36.781	40.289
23	18.137	22.337	27.141	32.007	35.172	38.076	41.638
24	19.037	23.337	28.241	33.196	36.415	39.364	42.980
25	19.939	24.337	29.339	34.382	37.652	40.646	44.314
26	20.843	25.336	30.435	35.563	38.885	41.923	45.642
27	21.749	26.336	31.528	36.741	40.113	43.195	46.963
28	22.657	27.336	32.620	37.916	41.337	44.461	48.278
29	23.567	28.336	33.711	39.087	42.557	45.722	49.588
30	24.478	29.336	34.800	40.256	43.773	46.979	50.892
40	33.660	39.335	45.616	51.805	55.758	59.342	63.691
50	42.942	49.335	56.334	63.167	67.505	71.420	76.154
75	66.417	74.334	82.858	91.061	96.217	100.839	106.393
100	90.133	99.334	109.141	118.498	124.342	129.561	135.807

参考文献

Abramowitz, M., and I.A. Stegun, eds.: *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables,* National Bureau of Standards, Washington, D.C. (1964).

Adiri, I., and B. Avi-Itzhak: A Time-Sharing Queue with a Finite Number of Customers, *J. Assoc. Comput. Mach., 16:* 315–323 (1969).

Ahrens, J.H., and U. Dieter: Computer Methods for Sampling from the Exponential and Normal Distributions, *Commun. Assoc. Comput. Mach., 15:* 873–882 (1972).

Ahrens, J.H., and U. Dieter: Computer Methods for Sampling from Gamma, Beta, Poisson, and Binomial Distributions, *Computing, 12:* 223–246 (1974).

Alexopoulos, C., C. Antonini, D. Goldsman, and M. Meterelliyoz: Performance of Folded Variance Estimators for Simulation, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 20,* No. 3, Article 11 (2010).

Alexopoulos, C., N.T. Argon, D. Goldsman, N.M. Steiger, G. Tokol, and J.R. Wilson: Efficient Computation of Overlapping Variance Estimators for Simulation, *INFORMS J. Comput., 19:* 314–327 (2007a).

Alexopoulos, C., N.T. Argon, D. Goldsman, G. Tokol, and J.R. Wilson: Overlapping Variance Estimators for Simulation, *Operations Res., 55:* 1090–1103 (2007b).

Alexopoulos, C., and D. Goldsman: To Batch or Not to Batch, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 14:* 76–114 (2004).

Alexopoulos, C., D. Goldsman, and R.F. Serfozo: Stationary Processes: Statistical Estimation, *Encyclopedia of Statistical Sciences,* 2d ed., N. Balakrishnan, C. Read, and B. Vidakovic, eds., John Wiley, New York (2006).

Alexopoulos, C., D. Goldsman, and J.R. Wilson: A New Perspective on Batched Quantile Estimation, *Proc. 2012 Winter Simulation Conference,* Berlin, Germany (2012).

Alexopoulos, C., and A.F. Seila: Output Data Analysis, in *Handbook of Simulation,* J. Banks, ed., John Wiley, New York (1998).

Alspaugh, C., T.A. Hepner, C. Tran, W. Youm, A.K. Legaspi, S. Ferenci, R.M. Fujimoto, and M. Choi: Navy NETWARS Interoperability Efforts, *Proceedings of the Spring 2004 Simulation Interoperability Workshop,* Arlington, Virginia (2004).

Anderson, T.W.: *The Statistical Analysis of Time Series,* John Wiley, New York (1994).

Anderson, T.W., and D.A. Darling: A Test of Goodness of Fit, *J. Am. Statist. Assoc., 49:* 765–769 (1954).

Andradóttir, S.: Simulation Optimization, in *Handbook of Simulation,* J. Banks, ed., John Wiley, New York (1998).

Andradóttir, S.: An Overview of Simulation Optimization via Random Search, in *Handbooks in Operations Research and Management Science: Simulation,* S.G. Henderson and B.L. Nelson, eds., Elsevier, New York (2006).

Andradóttir, S., and S.-H. Kim: Fully-Sequential Procedures for Comparing Constrained Systems Via Simulation, *Naval Res. Logistics, 57:* 403–421 (2010).

Andradóttir, S., and A.A. Prudius: Adaptive Random Search for Continuous Simulation Optimization, *Naval Res. Logistics, 57:* 583–604 (2010).

Andréasson, I.J.: Antithetic Methods in Queueing Simulations, Roy. Inst. Technol. Dept. Comput. Sci. Tech. Rep. NA 72.58, Stockholm (1972).

Angün, E., D. ven Hertog, G. Gürkan, and J.P.C. Kleijnen: Response Surface Methodology with Stochastic Constraints for Expensive Simulation, *J. Operational Res. Soc., 60:* 735–746 (2009).

Ankenman, B.E., and B.L. Nelson: A Quick Assessment of Input Uncertainty, *Proc. 2012 Winter Simulation Conference*, Berlin, Germany (2012).

Ankenman, B.E., B.L. Nelson, and J. Staum: Stochastic Kriging for Simulation Metamodeling, *Operations Res., 58:* 371–382 (2010).

Antonini, C., C. Alexopoulos, D. Goldsman, and J.R. Wilson: Area Variance Estimators for Simulation Using Folded Standardized Time Series, *IIE Trans., 41:* 134–144 (2009).

AnyLogic Company, The: AnyLogic simulation software, www.anylogic.com (2013).

Applied Materials, Inc.: AutoMod simulation software, www.automod.com (2013).
Argon, N.T., and S. Andradóttir: Replicated Batch Means for Steady-State Simulations, *Naval Res. Logistics, 53:* 508–524 (2006).
Argonne National Lab: Repast simulation software, repast.sourceforge.net (2013).
Arkin, B.L., and L.M. Leemis: Nonparametric Estimation of the Cumulative Intensity Function for a Nonhomogeneous Poisson Process from Overlapping Realizations, *Management Sci., 46:* 989–998 (2000).
Arnold, B.C.: A Note on Multivariate Distributions with Specified Marginals, *J. Am. Statist. Assoc., 62:* 1460–1461 (1967).
Artelli, M.J., and R.F. Deckro: Modeling the Lancaster Laws with System Dynamics, *J. of Defense Modeling and Simulation, 5:* 1–20 (2008).
Atkinson, A.C.: A Family of Switching Algorithms for the Computer Generation of Beta Random Variables, *Biometrika, 66:* 141–145 (1979a).
Atkinson, A.C.: The Computer Generation of Poisson Random Variables, *Appl. Statist., 28:* 29–35 (1979b).
Atkinson, A.C.: Recent Developments in the Computer Generation of Poisson Random Variables, *Appl. Statist., 28:* 260–263 (1979c).
Atkinson, A.C.: Tests of Pseudo-random Numbers, *Appl. Statist., 29:* 164–171 (1980).
Atkinson, A.C., and M.C. Pearce: The Computer Generation of Beta, Gamma, and Normal Random Variables, *J. Roy. Statist. Soc., A139:* 431–448 (1976).
Atkinson, A.C., and J. Whittaker: A Switching Algorithm for the Generation of Beta Random Variables with at Least One Parameter Less than 1, *J. Roy. Statist. Soc., A139:* 462–467 (1976).
Atkinson, A.C., and J. Whittaker: The Generation of Beta Random Variables with One Parameter Greater than and One Parameter Less than 1, *Appl. Statist., 28:* 90–93 (1979).
Averill M. Law & Associates, Inc.: *ExpertFit User's Guide,* Tucson, Arizona (2013).
Avramidis, A.N., and J.R. Wilson: A Splitting Scheme for Control Variates, *Operations Res. Letters, 14:* 187–198 (1993).
Avramidis, A.N., and J.R. Wilson: Integrated Variance Reduction Strategies for Simulation, *Operations Res., 44:* 327–346 (1996).
Avramidis, A.N., and J.R. Wilson: Correlation-Induction Techniques for Estimating Quantiles in Simulation Experiments, *Operations Res., 46:* 574–591 (1998).
Bäck, T.: *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, and Genetic Algorithms,* Oxford University Press, New York (1996).
Bäck, T., and H.-P. Schwefel: An Overview of Evolutionary Algorithms for Parameter Optimization, *Evolutionary Computation, 1:* 1–23 (1993).
Bagni, R., R. Berchi, and P. Cariello: A Comparison of Simulation Models Applied to Epidemics, *J. of Artificial Societies and Social Simulation, 5,* http://jasss.soc.surrey.ac.uk/5/3/5.html (June 2002).
Balci, O.: Verification, Validation and Testing, in *Handbook of Simulation,* J. Banks, ed., John Wiley, New York (1998).
Banks, J.: *Getting Started with AutoMod,* 2d ed., Brooks Automation, Inc., Chelmsford, Massachusetts (2004).
Banks, J., J.S. Carson, B.L. Nelson, and D.M. Nicol: *Discrete-Event System Simulation,* 5th ed., Prentice-Hall, Upper Saddle River, New Jersey (2010).
Barnett, V.: Probability Plotting Methods and Order Statistics, *Appl. Statist., 24:* 95–108 (1975).
Bartels, R.: The Rank Version of von Neumann's Ratio Test for Randomness, *J. Am. Statist. Assoc., 77:* 40–46 (1982).
Barton, R.R.: *Graphical Methods for Design of Experiments,* Springer, New York (1999).
Barton, R.R.: Simulation Optimization Using Metamodels, *Proc. 2009 Winter Simulation Conference,* Austin, Texas, pp. 230–238 (2009).
Barton, R.R.: Simulation Experiment Design, *Proc. 2010 Winter Simulation Conference*, Baltimore, Maryland, pp. 75–86 (2010).
Barton, R.R.: Tutorial: Input Uncertainty in Output Analysis, *Proc. 2012 Winter Simulation Conference,* Berlin, Germany (2012).
Barton, R.R., and M. Meckesheimer: Metamodel-Based Simulation Optimization, in *Handbooks in Operations Research and Management Science: Simulation,* S.G. Henderson and B.L.

Nelson, eds., Elsevier, New York (2006).
Barton, R.R., B.L. Nelson, and W. Xie: Quantifying Input Uncertainty via Simulation Confidence Intervals, to appear in *INFORMS J. Comput.* (2013).
Barton, R.R., and L.W. Schruben: Uniform and Bootstrap Resampling of Empirical Distributions, *Proc. 1993 Winter Simulation Conference,* Los Angeles, pp. 503–508 (1993).
Barton, R.R., and L.W. Schruben: Resampling Methods for Input Modeling, *Proc. 2001 Winter Simulation Conference,* Washington, D.C., pp. 372–378 (2001).
Bauer, K.W.: Control Variate Selection for Multiresponse Simulation, Ph.D. Dissertation, School of Industrial Engineering, Purdue University, West Lafayette, Indiana (1987).
Bauer, K.W., and J.R. Wilson: Control-Variate Selection Criteria, *Naval Res. Logist., 39*: 307–321 (1992).
Bays, C., and S.D. Durham: Improving a Poor Random Number Generator, *Assoc. Comput. Mach. Trans. Math. Soft., 2:* 59–64 (1976).
Beaverstock, M.C., A. Greenwood, E. Lavery, and W.B. Nordgren: *Applied Simulation: Modeling and Analysis Using FlexSim,* 3d ed., FlexSim Software Products, Orem, Utah (2013).
Bechhofer, R.E., T.J. Santner, and D. Goldsman: *Design and Analysis for Statistical Selection, Screening and Multiple Comparisons,* John Wiley, New York (1995).
Beckman, R.J., and G.L. Tietjen: Maximum Likelihood Estimation for the Beta Distribution, *J. Statist. Comput. Simul., 7:* 253–258 (1978).
Beeker, E.R., and E.H. Page: A Case Study of the Development and Use of a MANA-Based Federation for Studying U.S. Border Operations, *Proc. of the 2006 Winter Simulation Conference,* pp. 841–847, Monterey, California (2006).
Bertsekas, D.P., and R. Gallager: *Data Networks,* 2d ed., Prentice-Hall, Englewood Cliffs, New Jersey (1992).
Best, D.J., and D.E. Roberts: The Percentage Points of the χ^2 Distribution, *Appl. Statist., 24:* 385–388 (1975).
Bettonvil, B., and J.P.C. Kleijnen, Searching for Important Factors in Simulation Models with Many Factors: Sequential Bifurcation, *Eur. J. Operational Res., 96:* 180–194 (1997).
Bhattacharjee, G.P.: Algorithm AS32: The Incomplete Gamma Integral, *Appl. Statist., 19:* 285–287 (1970).
Biller, B.: Copula-Based Multivariate Input Models for Stochastic Simulation, *Operations Res., 57:* 878–892 (2009).
Biller, B., and S. Ghosh: Multivariate Input Processes, in *Handbooks in Operations Research and Management Science: Simulation,* S.G. Henderson and B.L. Nelson, eds., Elsevier, New York (2006).
Biller, B., and B.L. Nelson: Modeling and Generating Multivariate Time-Series Input Processes Using a Vector Autoregressive Technique, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 13:* 211–237 (2003).
Biller, B., and B.L. Nelson: Fitting Times-Series Input Processes for Simulation, *Operations Res., 53:* 549–559 (2005).
Biller, B., and B.L. Nelson: Evaluation of the ARTAFIT Method for Fitting Time-Series Input Processes for Simulation, *INFORMS J. Comput., 20:* 485–498 (2008).
Billingsley, P.: *Convergence of Probability Measures,* 2d ed., John Wiley, New York (1999).
Billingsley, P., D.J. Croft, D.V. Huntsberger, and C.J. Watson: *Statistical Inference for Management and Economics,* 3d ed., Allyn & Bacon, Boston (1986).
Bischak, D.P., W.D. Kelton, and S.M. Pollack: Weighted Batch Means for Confidence Intervals in Steady-State Simulations, *Management Sci., 39:* 1002–1019 (1993).
Bishop, C.M.: *Neural Networks for Pattern Recognition,* Oxford University Press, New York (1995).
Blomqvist, N.: On the Transient Behaviour of the GI/G/1 Waiting-Times, *Skand. Aktuarietidskr.,* 118–129 (1970).
Blum, C., and A. Roli (2003): Metaheuristics and Combinatorial Optimization: Overview and Conceptual Comparison, *Assoc. Comput. Mach. Comp. Surveys, 5:* 208–308 (2003).
Bodoh, D.J., and F. Wieland: Performance Experiments with the High Level Architecture and the Total Airport and Airspace Model (TAAM), *Proc. of the 17th Workshop on Parallel and Distributed Simulation,* San Diego, pp. 31–39 (2003).
Boesel, J., B.L. Nelson, and N. Ishii: A Framework for Simulation-Optimization Software, *IIE*

Trans., 35: 221–229 (2003).
Boesel, J., B.L. Nelson, and S.-H. Kim: Using Ranking and Selection to "Clean Up" after Simulation Optimization, *Operations Res., 51:* 814–825 (2003).
Bonabeau, E.: Agent-Based Modeling: Methods and Techniques for Simulating Human Systems, *Proc. of the National Academy of Sciences, 99:* 7280–7287 (May 2002).
Bonabeau, E., and C. Meyer: Swarm Intelligence: A Whole New Way to Think about Business, *Harvard Business Review,* 107–114 (May 2001).
Borshchev, A.: Personal Communication (2013).
Bosten, N.E., and E.L. Battiste: Remark on Algorithm 179, *Commun. Assoc. Comput. Mach., 17:* 156–157 (1974).
Box, G.E.P., and N.R. Draper: *Response Surfaces, Mixtures, and Ridge Analyses,* 2d ed., John Wiley, New York (2008).
Box, G.E.P., J.S. Hunter, and W.G. Hunter: *Statistics for Experimenters: Design, Innovation, and Discovery,* 2d ed., John Wiley, Hoboken, New Jersey (2005).
Box, G.E.P., G.M. Jenkins, and G.C. Reinsel: *Time Series Analysis: Forecasting and Control,* 4th ed., John Wiley, New York (2009).
Box, G.E.P., and M.E. Muller: A Note on the Generation of Random Normal Deviates, *Ann. Math. Statist., 29:* 610–611 (1958).
Bratley, P., B.L. Fox, and L.E. Schrage: *A Guide to Simulation,* 2d ed., Springer-Verlag, New York (1987).
Braun, M.: *Differential Equations and Their Applications,* Applied Mathematical Sciences, Vol. 15, Springer-Verlag, New York (1975).
Breiman, L.: *Statistics: With a View Toward Applications,* Houghton Mifflin, Boston (1973).
Brown, L.P., T.M. Cioppa, and T.W. Lucas: Agent-Based Simulations Supporting Military Analysis, *Phalanx, 37:* 29–32 (September 2004).
Brown, R.: Calendar Queues: A Fast O(1) Priority Queue Implementation for the Simulation Event Set Problem, *Commun. Assoc. Comput. Mach., 31:* 1220–1227 (1988).
Bryant, R.E.: Simulation of Packet Communication Architecture Computer Systems, *MIT-LCSTR-188,* Massachusetts Institute of Technology, Cambridge, Massachusetts (1977).
Burt, J.M., Jr., and M.B. Garman: Conditional Monte Carlo: A Simulation Technique for Stochastic Network Analysis, *Management Sci., 18:* 207–217 (1971).
Burt, J.M., Jr., D.P. Gaver, and M. Perlas: Simple Stochastic Networks: Some Problems and Procedures, *Naval Res. Logist. Quart., 17:* 439–459 (1970).
Buss, A.H.: Modeling with Event Graphs, *Proc. 1996 Winter Simulation Conference,* San Diego, pp. 153–160 (1996).
Buss, A., and A.A. Rowaei: A Comparison of Discrete Event and Discrete Time, *Proc. of the 2010 Winter Simulation Conference*, pp. 1468–1477, Baltimore, Maryland (2010).
Campolongo, F., J.P.C. Kleijnen, and T. Andres: Screening Methods, in *Sensitivity Analysis,* A. Saltelli, K. Chan, and E.M. Scott, eds., John Wiley, New York (2000).
Cario, M.C., and B.L. Nelson: Autoregressive to Anything: Time-Series Input Processes for Simulation, *Operations Res. Letters, 19:* 51–58 (1996).
Cario, M.C., and B.L. Nelson: Numerical Methods for Fitting and Simulating Autoregressive-To-Anything Processes, *INFORMS J. Comput., 10:* 72–81 (1998).
Cario, M.C., B.L. Nelson, S.D. Roberts, and J.R. Wilson: Modeling and Generating Random Vectors with Arbitrary Marginal Distributions and Correlation Matrix, Technical Report, Dept. of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois (2002).
Carley, K.M., D.B. Fridsma, E. Casman, A. Yahja, N. Altman, L.-C. Chen, B. Kaminsky, and D. Nave: Biowar: Scalable Agent-Based Model of Bioattacks, *IEEE Trans. on Systems, Man, and Cybernetics, Part A, 36:* 252–265 (2006).
Carson, J.S.: Variance Reduction Techniques for Simulated Queueing Processes, Univ. Wisconsin Dept. Ind. Eng. Tech. Rep. 78-8, Madison, Wisconsin (1978).
Carson J.S.: Convincing Users of Model's Validity Is Challenging Aspect of Modeler's Job, *Ind. Eng., 18:* 74–85 (June 1986).
Carson, J.S.: Model Verification and Validation, *Proc. 2002 Winter Simulation Conference,* San Diego, pp. 42–58 (2002).

Carson, J.S., and A.M. Law: Conservation Equations and Variance Reduction in Queueing Simulations, *Operations Res., 28:* 535–546 (1980).

Carson, J.S., N. Wilson, D. Caroll, and C.H. Wysocki: A Discrete Simulation Model of a Cigarette Fabrication Process, *Proc. Twelfth Modeling and Simulation Conference,* University of Pittsburgh, pp. 683–689 (1981).

Carter, G., and E.J. Ignall: Virtual Measures: A Variance Reduction Technique for Simulation, *Management Sci., 21:* 607–616 (1975).

Chandra, M., N.D. Singpurwalla, and M.A. Stephens: Kolmogorov Statistics for Tests of Fit for the Extreme-Value and Weibull Distributions, *J. Am. Statist. Assoc., 76:* 729–731 (1981).

Chandy, K.M., and J. Misra: Distributed Simulation: A Case Study in Design and Verification of Distributed Programs, *IEEE Trans. Software Eng., SE-5:* 440–452 (1979).

Chang, K.-H., L.J. Hong, and H. Wan: Stochastic Trust-Region Response-Surface Method (STRONG)—A New Response-Surface Framework for Simulation Optimization, *INFORMS J. Comput., 25:* 230–243 (2013).

Charnes, J.S.: Analyzing Multivariate Output, *Proc. 1995 Winter Simulation Conference,* Washington, D.C., pp. 201–208 (1995).

Chen, B.-C., and R.G. Sargent: Using Standardized Time Series to Estimate the Difference between Two Stationary Stochastic Processes, *Operations Res., 35:* 428–436 (1987).

Chen, C.-H.: A Lower Bound for the Correct Subset-Selection Probability and Its Application to Discrete-Event System Simulations, *IEEE Trans. Auto. Cont., 41:* 1227–1231 (1996).

Chen, C.-H., D. He, M. Fu, and L.H. Lee: Efficient Simulation Budget Allocation for Selecting an Optimal Subset, *INFORMS J. Comput., 20:* 579–595 (2008).

Chen, C.-H., and L.H. Lee: Stochastic Simulation Optimization: An Optimal Computing Budget Allocation, World Scientific Publishing, Singapore (2011).

Chen, C.-H., J. Lin, E. Yücesan, and S.E. Chick: Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization, *Disc. Event Dyn. Syst: Theory App., 10:* 251–270 (2000).

Chen, C.-H., E. Yücesan, L. Dai, and H.C. Chen: Optimal Budget Allocation for Discrete-Event Simulation Experiments, *IIE Trans., 42:* 60–70 (2010).

Chen, E.J.: Subset Selection Procedures, *J. of Simulation, 3:* 202–210 (2009).

Chen, E.J., and W.D. Kelton: Sequential Selection Procedures: Using Sample Means to Improve Efficiency, *Eur. J. Operational Research, 166:* 133–153 (2005).

Chen, E.J., and W.D. Kelton: Quantile and Tolerance-Interval Estimation in Simulation, *Eur. J. Operational Res., 168:* 520–540 (2006).

Chen, E.J., and W.D. Kelton: Confidence Interval Estimation Using Quasi-Independent Sequences, *IIE Trans., 42:* 83–93 (2009).

Chen, H.: Initialization for NORTA: Generation of Random Vectors with Specified Marginals and Correlations, *INFORMS J. Comput., 13:* 312–331 (2001).

Chen, H., and Y. Asau: On Generating Random Variates from an Empirical Distribution, *AIIE Trans., 6:* 163–166 (1974).

Chen, P.: On Selecting the Best of *k* Systems: An Expository Survey of Subset-Selection Multinomial Procedures, *Proc. 1988 Winter Simulation Conference,* San Diego, pp. 440–444 (1988).

Chen, X., B.E. Ankenman, and B.L. Nelson: The Effects of Common Random Numbers on Stochastic Kriging Metamodels, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 22,* No. 2, Article 7 (2010).

Cheng, R.C.H.: The Generation of Gamma Variables with Non-integral Shape Parameter, *Appl. Statist., 26:* 71–75 (1977).

Cheng, R.C.H.: Generating Beta Variates with Nonintegral Shape Parameters, *Commun. Assoc. Comput. Mach., 21:* 317–322 (1978).

Cheng, R.C.H.: The Use of Antithetic Variates in Computer Simulations, *J. Operational Res. Soc., 33:* 229–237 (1982).

Cheng, R.C.H.: Antithetic Variate Methods for Simulation of Processes with Peaks and Troughs, *Eur. J. Operational Res., 15:* 227–236 (1984).

Cheng, R.C.H.: Searching for Important Factors: Sequential Bifurcation under Uncertainty, *Proc.*

1997 Winter Simulation Conference, Atlanta, pp. 275–280 (1997).
Cheng, R.C.H.: Resampling Methods, in *Handbooks in Operations Research and Management Science: Simulation,* S.G. Henderson and B.L. Nelson, eds., Elsevier, New York (2006).
Cheng, R.C.H., and N.A.K. Amin: Estimating Parameters in Continuous Univariate Distributions with a Shifted Origin, *J. Roy. Statist. Soc. B, 45:* 394–403 (1983).
Cheng, R.C.H., and C.S.M. Currie: Resampling Methods of Analysis in Simulation Studies, *Proc. of the 2009 Winter Simulation Conference*, pp. 45–59, Austin, Texas (2009).
Cheng, R.C.H., and G.M. Feast: Some Simple Gamma Variate Generators, *Appl. Statist., 28:* 290–295 (1979).
Cheng, R.C.H., and G.M. Feast: Control Variates with Known Mean and Variance, *J. Operational Res. Soc., 31:* 51–56 (1980).
Cheng, R.C.H., and W. Holland: Sensitivity of Computer Simulation Experiments to Errors in Input Data, *J. Statist. Comput. Simul., 57:* 219–241 (1997).
Cheng, R.C.H., and W. Holland: Two-Point Methods for Assessing Variability in Simulation Output, *J. Statist. Comput. Simul., 60:* 183–205 (1998).
Cheng, R.C.H., and W. Holland: Calculation of Confidence Intervals for Simulation Output, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 14:* 344–362 (2004).
Cheng, R.C.H., and J.P.C. Kleijnen: Improved Design of Queueing Simulation Experiments with Highly Heteroscedastic Responses, *Operations Res., 47:* 762–777 (1999).
Chernoff, H., and E.L. Lehmann: The Use of Maximum Likelihood Estimates in χ^2 Tests for Goodness of Fit, *Ann. Math. Statist., 25:* 579–586 (1954).
Chick, S.E.: Selecting the Best System: A Decision-Theoretic Approach, *Proc. 1997 Winter Simulation Conference,* Atlanta, pp. 326–333 (1997).
Chick, S.E.: Input Distribution Selection for Simulation Experiments: Accounting for Input Uncertainty, *Operations Res., 49:* 744–758 (2001).
Chick, S.E.: Subjective Probability and Bayesian Methodology, in *Handbooks in Operations Research and Management Science: Simulation,* S.G. Henderson and B.L. Nelson, eds., Elsevier, New York (2006).
Chick, S.E., and P. Frazier: Sequential Sampling with Economics of Selection Procedures, *Management Sci., 58:* 550–569 (2012).
Chick, S.E., and N. Gans: Economic Analysis of Simulation Selection Problems, *Management Sci., 55:* 421–437 (2009).
Chick, S.E., and K. Inoue: New Procedures to Select the Best Simulated System Using Common Random Numbers, *Management Sci., 47:* 1133–1149 (2001a).
Chick, S.E., and K. Inoue: New Two-Stage and Sequential Procedures for Selecting the Best Simulated System, *Operations Res., 49:* 732–744 (2001b).
Choi, S.C., and R. Wette: Maximum Likelihood Estimation of the Parameters of the Gamma Distribution and Their Bias, *Technometrics, 11:* 683–690 (1969).
Chow, Y.S., and H. Robbins: On the Asymptotic Theory of Fixed-Width Sequential Confidence Intervals for the Mean, *Ann. Math. Statist., 36:* 457–462 (1965).
Chung, K., J. Sang, and V. Rego: A Performance Comparison of Event Calendar Algorithms: An Empirical Approach, *Software—Practice and Experience, 23:* 1107–1138 (1993).
Chung, K.L.: *A Course in Probability Theory,* 2d ed., Academic Press, New York (1974).
Çinlar, E.: *Introduction to Stochastic Processes,* Prentice-Hall, Englewood Cliffs, New Jersey (1975).
Cioppa, T.M., and T.W. Lucas: Efficient Nearly Orthogonal and Space-Filling Latin Hypercubes, *Technometrics, 49:* 45–55 (2007).
Cioppa, T.M., T.W. Lucas, and S.M. Sanchez: Military Applications of Agent-Based Simulations, *Proc. of the 2004 Winter Simulation Conference,* pp. 171–180, Washington, D.C. (2004).
Cohen, A.C., and B.J. Whitten: Estimation in the Three-Parameter Lognormal Distribution, *J. Am. Statist. Assoc., 75:* 399–404 (1980).
Collier, N.T., and M.J. North: Parallel Agent-Based Simulation with Repast for High Performance Computing, to appear in *SIMULATION* (2013).
Collins, B.J.: Compound Random Number Generators, *J. Am. Statist. Assoc., 82:* 525–527 (1987).
Comfort, J.C.: The Simulation of a Microprocessor-Based Event Set Processor, *Proc. 14th Annual*

Simulation Symposium, pp. 17–33 (1981).

Conover, W.J.: *Practical Nonparametric Statistics,* 3d ed., John Wiley, New York (1999).

Coveyou, R.R., and R.D. MacPherson: Fourier Analysis of Uniform Random Number Generators, *J. Assoc. Comput. Mach., 14:* 100–119 (1967).

Cran, G.W., K.J. Martin, and G.E. Thomas: A Remark on Algorithm AS63: The Incomplete Beta Integral, AS64: Inverse of the Incomplete Beta Function Ratio, *Appl. Statist., 26:* 111–114 (1977).

Crane, M.A., and D.L. Iglehart: Simulating Stable Stochastic Systems, I: General Multiserver Queues, *J. Assoc. Comput. Mach., 21:* 103–113 (1974a).

Crane, M.A., and D.L. Iglehart: Simulating Stable Stochastic Systems, II: Markov Chains, *J. Assoc. Comput. Mach., 21:* 114–123 (1974b).

Crane, M.A., and D.L. Iglehart: Simulating Stable Stochastic Systems, III: Regenerative Processes and Discrete-Event Simulations, *Operations Res., 23:* 33–45 (1975).

Crane, M.A., and A.J. Lemoine: *An Introduction to the Regenerative Method for Simulation Analysis,* Lecture Notes in Control and Information Sciences, Vol. 4, Springer-Verlag, New York (1977).

Cressie, N.A.C.: *Statistics for Spatial Data,* Revised Edition, John Wiley, New York (1993).

D'Agostino, R.B., and M.A. Stephens: *Goodness-of-Fit Tests,* Marcel Dekker, New York (1986).

Dagpunar, J.: *Principles of Random Variate Generation,* Clarendon Press, Oxford, England (1988).

Dahmann, J.S., R.M. Fujimoto, and R.M. Weatherly: The DoD High Level Architecture: An Update, *Proc. 1998 Winter Simulation Conference,* pp. 797–804, Washington, D.C. (1998).

Daley, D.J.: The Serial Correlation Coefficients of Waiting Times in a Stationary Single Server Queue, *J. Austr. Math. Soc., 8:* 683–699 (1968).

Damerdji, H.: Strong Consistency and Other Properties of the Spectral Variance Estimator, *Management Sci., 37:* 1424–1440 (1991).

Damerdji, H.: Strong Consistency of the Variance Estimator in Steady-State Simulation Output Analysis, *Math. of Operations Res., 19:* 494–512 (1994).

Damerdji, H., and M.K. Nakayama: Two-Stage Multiple Comparison Procedures for Steady-State Simulations, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 9:* 1–30 (1999).

Dean, A.M., and S.M. Lewis: *Screening: Methods for Experimentation in Industry, Drug Discovery, and Genetics,* Springer, New York (2006).

Debuse, J.C.W., V.J. Rayward-Smith, and G.D. Smith: Parameter Optimisation for a Discrete Event Simulator, *J. Comp. and Indust. Eng., 37:* 181–184 (1999).

DeGroot, M.H.: *Probability and Statistics,* Addison-Wesley, Reading, Massachusetts (1975).

DeRiggi, D.F.: Unimodality of Likelihood Functions for the Binomial Distribution, *J. Am. Statist. Assoc., 78:* 181–183 (1983).

Devore, J.L.: *Probability and Statistics for Engineering and the Sciences,* 7th ed., Brooks/Cole, Belmont, California (2008).

Devroye, L.: The Computer Generation of Poisson Random Variables, *Computing, 26:* 197–207 (1981).

Devroye, L.: *Non-Uniform Random Variate Generation,* Springer-Verlag, New York (1986).

Devroye, L.: Generating Sums in Constant Average Time, *Proc. 1988 Winter Simulation Conference,* San Diego, pp. 425–431 (1988).

Devroye, L.: Random Variate Generation for Multivariate Unimodal Densities, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 7:* 447–477 (1997).

Donohue, J.M., E.C. Houck, and R.H. Myers: Simulation Designs for Quadratic Response Surface Models in the Presence of Model Misspecification, *Management Sci., 38:* 1765–1791 (1992).

Donohue, J.M., E.C. Houck, and R.H. Myers: Simulation Designs for the Estimation of Quadratic Response Surface Gradients in the Presence of Model Misspecification, *Management Sci., 41:* 244–262 (1995).

Dubey, S.D.: On Some Permissible Estimators of the Location Parameter of the Weibull and Certain Other Distributions, *Technometrics, 9:* 293–307 (1967).

Dudewicz, E.J.: Random Numbers: The Need, the History, the Generators, in *Statistical Distributions in Scientific Work 2,* G.P. Patil, S. Kotz, and J.K. Ord, eds., D. Reidel, Dordrecht, The Netherlands (1975). [Also reprinted in *Modern Design and Analysis of Discrete-Event Com-*

puter Simulations, E.J. Dudewicz and Z.A. Karian, eds., IEEE Computer Society (1985).]
Dudewicz, E.J., and S.R. Dalal: Allocation of Observations in Ranking and Selection with Unequal Variances, *Sankhya, B37:* 28–78 (1975).
Durbin, J.: Kolmogorov-Smirnov Tests When Parameters Are Estimated with Applications to Tests of Exponentiality and Tests on Spacings, *Biometrika, 62:* 5–22 (1975).
Economist, The: Software: Simulating the Behaviour of Crowds of People, or Swarms of Animals, Has Both Frivolous and Important Uses, www.economist.com/node/13174313 (March 7, 2009).
Economist, The: Agents of Change, www.economist.com/node/16636121 (July 24, 2010).
Efron, B., and R.J. Tibshirani: *Introduction to the Bootstrap,* Chapman and Hall, New York (1993).
Eglese, R.W.: Simulated Annealing: A Tool for Operational Research, *Eur. J. Operational Res., 46:* 271–281 (1990).
Epstein, J.M.: *Generative Social Science: Studies in Agent-Based Computational Modeling*, Princeton University Press, Princeton, New Jersey (2007).
Epstein, J.M.: Modeling to Contain Pandemics, *Nature, 460:* 687 (August 2009).
Epstein, J.M., and R. Axtell: *Growing Artificial Societies: Social Science from the Bottom Up,* MIT Press, Cambridge, Massachusetts (1996).
Evans, J.R., and D.L. Olson: *Introduction to Simulation and Risk Analysis,* 2d ed., Prentice Hall, Englewood Cliffs, New Jersey (2002).
Feltner, C.E., and S.A. Weiner: Models, Myths and Mysteries in Manufacturing, *Ind. Eng., 17:* 66–76 (July 1985).
Filliben, J.J.: The Probability Plot Correlation Coefficient Test for Normality, *Technometrics, 17:* 111–117 (1975).
Fishman G.S.: *Spectral Methods in Econometrics,* Harvard University Press, Cambridge, Massachusetts (1969).
Fishman, G.S.: Estimating Sample Size in Computer Simulation Experiments, *Management Sci., 18:* 21–38 (1971).
Fishman, G.S.: Bias Considerations in Simulation Experiments, *Operations Res., 20:* 785–790 (1972).
Fishman, G.S.: *Concepts and Methods in Discrete Event Digital Simulation,* John Wiley, New York (1973a).
Fishman, G.S.: Statistical Analysis for Queueing Simulations, *Management Sci., 20:* 363–369 (1973b).
Fishman, G.S.: Estimation in Multiserver Queueing Simulations, *Operations Res., 22:* 72–78 (1974).
Fishman, G.S.: Achieving Specific Accuracy in Simulation Output Analysis, *Commun. Assoc. Comput. Mach., 20:* 310–315 (1977).
Fishman, G.S.: *Principles of Discrete Event Simulation,* John Wiley, New York (1978).
Fishman, G.S.: *Monte Carlo: Concepts, Algorithms, and Applications,* Springer Verlag, New York (1996).
Fishman, G.S.: *Discrete-Event Simulation,* Springer, New York (2001).
Fishman, G.S.: *A First Course in Monte Carlo,* Duxbury Press, Pacific Grove, California (2006).
Fishman, G.S., and B.D. Huang: Antithetic Variates Revisited, *Commun. Assoc. Comput. Mach., 26:* 964–971 (1983).
Fishman, G.S., and P.J. Kiviat: The Analysis of Simulation-Generated Time Series, *Management Sci., 13:* 525–557 (1967).
Fishman, G.S., and P.J. Kiviat: The Statistics of Discrete-Event Simulation, *Simulation, 10:* 185–195 (1968).
Fishman, G.S., and L.R. Moore: A Statistical Evaluation of Multiplicative Congruential Random Number Generators with Modulus $2^{31} - 1$, *J. Am. Statist. Assoc., 77:* 129–136 (1982).
Fishman, G.S., and L.R. Moore: Sampling from a Discrete Distribution While Preserving Monotonicity, *Am. Statistician, 38:* 219–223 (1984).
Fishman, G.S., and L.R. Moore: An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus $2^{31} - 1$, *SIAM J. Sci. Stat. Comput., 7:* 24–45 (1986).
Fishman, G.S., and L.S. Yarberry: An Implementation of the Batch Means Method, *INFORMS*

J. Comput., 9: 296–310 (1997).

FlexSim Software Products, Inc.: FlexSim Healthcare simulation software, www.flexsim.com (2013).

FlexSim Software Products, Inc.: FlexSim simulation software, www.flexsim.com (2013).

Forbes, C., M. Evans, N. Hastings, and B. Peacock: *Statistical Distributions,* 4th ed., John Wiley, New York (2011).

Forsythe, G.E.: von Neumann's Comparison Method for Random Sampling from the Normal and Other Distributions, *Math. Comput., 26:* 817–826 (1972).

Fossett, F.A., D. Harrison, H. Weintrob, and S.I. Gass: An Assessment Procedure for Simulation Models: A Case Study, *Operations Res., 39:* 710–723 (1991).

Fox, B.L., D. Goldsman, and J.J. Swain: Spaced Batch Means, *Operations Res. Letters, 10:* 255–263 (1991).

Franklin, W.W., and K.P. White: Stationarity Tests and MSER-5: Exploring the Intuition Behind Mean-Squared-Error-Reduction in Detecting and Correcting Initialization Bias, *Proc. 2008 Winter Simulation Conference*, Miami, Florida, pp. 541–546 (2008).

Franta, W.R.: A Note on Random Variate Generators and Antithetic Sampling, *INFOR, 13:* 112–117 (1975).

Friedman, L.W.: *The Simulation Metamodel,* Kluwer Academic Publishers, Dordrecht, The Netherlands (1996).

Fu, M.C.: Optimization for Simulation: Theory vs. Practice, *INFORMS J. Comput., 14:* 192–215 (2002).

Fu, M.C.: Gradient Estimation, in *Handbooks in Operations Research and Management Science: Simulation,* S.G. Henderson and B.L. Nelson, eds., Elsevier, New York (2006).

Fu, M.C.: What You Should Know about Simulation and Derivatives, *Naval Res. Logistics, 55:* 723–736 (2008).

Fu, M.C.: *Handbook on Simulation Optimization*, Springer, New York (2013).

Fu, M.C., F.W. Glover, and J. April: Simulation Optimization: A Review, New Developments, and Applications, *Proc. 2005 Winter Simulation Conference,* Orlando (2005).

Fu, M.C., J.H. Hu, C.-H. Chen, and X. Xiong: Simulation Allocation for Determining the Best Design in the Presence of Correlated Sampling, *INFORMS J. Comput., 19:* 101–111 (2007).

Fujimoto, R.M.: Parallel and Distributed Simulation, in *Handbook of Simulation,* J. Banks, ed., John Wiley, New York (1998).

Fujimoto, R.M.: *Parallel and Distributed Simulation Systems,* John Wiley, New York (2000).

Fujimoto, R.M.: *Distributed Simulation Systems, Proc. 2003 Winter Simulation Conference,* New Orleans, pp. 124–134 (2003).

Fujimoto, R.M., K. Perumalla, A. Park, H. Wu, M.H. Ammar, and G.F. Riley: Large-Scale Network Simulation: How Big? How Fast?, *Proc. of the 11th International Workshop on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems,* pp. 116–125, Orlando (2003).

Fushimi, M.: Random Number Generation with the Recursion $X_t = X_{t-3p} \oplus X_{t-3q}$, *J. of Comput. and App. Math., 31:* 105–118 (1990).

Gafarian, A.V., C.J. Ancker, Jr., and F. Morisaku: Evaluation of Commonly Used Rules for Detecting "Steady-State" in Computer Simulation, *Naval Res. Logist. Quart., 25:* 511–529 (1978).

Gal, S., R.Y. Rubinstein, and A. Ziv: On the Optimality and Efficiency of Common Random Numbers, *Math. Comput. Simul., 26:* 502–512 (1984).

Gardner, M.: The Fantastic Combinations of John Conway's New Solitaire Game "Life," *Scientific American, 223:* 120–123 (1970).

Garman, M.B.: More on Conditioned Sampling in the Simulation of Stochastic Networks, *Management Sci., 19:* 90–95 (1972).

Gass, S.I.: Decision-Aiding Models Validation, Assessment, and Related Issues in Policy Analysis, *Operations Res., 31:* 603–631 (1983).

Gass, S.I., and B.W. Thompson: Guidelines for Model Evaluation: An Abridged Version of the U.S. General Accounting Office Exposure Draft, *Operations Res., 28:* 431–439 (1980).

Gaver, D.P., and G.S. Shedler: Control Variable Methods in the Simulation of a Model of a Multiprogrammed Computer System, *Naval Res. Logist. Quart., 18:* 435–450 (1971).

Gaver, D.P., and G.L. Thompson: *Programming and Probability Models,* Wadsworth, Monterey, California (1973).

Gebhardt, F.: Generating Pseudo-random Numbers by Shuffling a Fibonacci Sequence, *Math. Comput., 21:* 708–709 (1967).

Gentle, J. E.: *Random Number Generation and Monte Carlo Methods,* 3d ed., Springer, New York (2010).

George, L.L.: Variance Reduction for a Replacement Process, *Simulation, 29:* 65–74 (1977).

Gerhardt, I., and B.L. Nelson: Transforming Renewal Processes for Simulation of Nonstationary Arrival Processes, *INFORMS J. Comput., 21:* 630–640 (2009).

Ghosh, S., and S.G. Henderson: Chessboard Distributions and Random Vectors with Specified Marginals and Covariance Matrix, *Operations Res., 50:* 820–834 (2002).

Ghosh, S., and S.G. Henderson: Behavior of the NORTA Method for Correlated Random Vector Generation as the Dimension Increases, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul.,13:* 276–294 (2003).

Ghosh, S., and S.G. Henderson: Corrigendum: Behavior of the NORTA Method for Correlated Random Vector Generation as the Dimension Increases, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 19,* No. 4, Article 20 (2009).

Gibbons, J.D.: *Nonparametric Methods for Quantitative Analysis,* 2d ed., American Sciences Press, Columbus, Ohio (1985).

Gilbert, N., and K.G. Troitzsch: *Simulation for the Social Scientist,* 2d ed., Open University Press, Maidenhead, England (2005).

Glasserman, P.: *Gradient Estimation via Perturbation Analysis,* Kluwer Academic Publishers, Boston (1991).

Glasserman, P: *Monte Carlo Methods in Financial Engineering,* Springer, New York (2004).

Glasserman, P., P. Heidelberger, P. Shahabuddin, and T. Zajic: Multilevel Splitting for Estimating Rare Event Probabilities, *Operations Res., 47:* 585–600 (1999).

Glasserman, P., and T.-W. Liu: Rare-Event Simulation for Multistage Production-Inventory Systems, *Management Sci., 42:* 1291–1307 (1996).

Glasserman, P., and D.D. Yao: Some Guidelines and Guarantees for Common Random Numbers, *Management Sci., 38:* 884–908 (1992).

Gleser, L.J.: Exact Power of Goodness-of-Fit Tests of Kolmogorov Type for Discontinuous Distributions, *J. Am. Statist. Assoc., 80:* 954–958 (1985).

Glover, F., and M. Laguna: *Tabu Search,* Kluwer Academic Publishers, New York (1997).

Glover, F., and M. Laguna: Tabu Search, in the *Handbook of Applied Optimization,* pp. 194–208, P.M. Pardalos and M.G.C. Resende, eds., Oxford University Press, New York (2002).

Glynn, P.W.: A Non-Rectangular Sampling Plan for Estimating Steady-State Means, *Proc. of the 6th Army Conference on Applied Mathematics and Computing,* pp. 965–978 (1988).

Glynn, P.W.: Some New Results on the Initial Transient Problem, *Proc. 1995 Winter Simulation Conference,* Arlington, Virginia, pp. 165–170 (1995).

Glynn, P.W., and D.L. Iglehart: The Theory of Standardized Time Series, *Math. Operations Res., 15:* 1–16 (1990).

Glynn, P.W., and W. Whitt: Indirect Estimation via $L = \lambda w$, *Operations Res., 37:* 82–103 (1989).

Glynn, P.W., and W. Whitt: The Asymptotic Efficiency of Simulation Estimators, *Operations Res., 40:* 505–520 (1992a).

Glynn, P.W., and W. Whitt: The Asymptotic Validity of Sequential Stopping Rules for Stochastic Simulations, *Ann. of Applied Probability, 2:* 180–198 (1992b).

Gnanadesikan, R., R.S. Pinkham, and L.P. Hughes: Maximum Likelihood Estimation of the Parameters of the Beta Distribution from Smallest Order Statistics, *Technometrics, 9:* 607–620 (1967).

Goldsman, D.: On Selecting the Best of k Systems: An Expository Survey of Indifference-Zone Multinomial Procedures, *Proc. 1984 Winter Simulation Conference,* Dallas, pp. 107–112 (1984a).

Goldsman, D.: A Multinomial Ranking and Selection Procedure: Simulation and Applications, *Proc. 1984 Winter Simulation Conference,* Dallas, pp. 259–264 (1984b).

Goldsman, D., S.-H. Kim, W.S. Marshall, and B.L. Nelson: Ranking and Selection for Steady-State Simulation: Procedures and Perspectives, *INFORMS J. Comput., 14:* 2–19 (2002).

Goldsman, D., M. Meketon, and L.W. Schruben: Properties of Standardized Time Series Weighted Area Variance Estimators, *Management Sci., 36:* 602–612 (1990).

Goldsman, D., and B.L. Nelson: Comparing Systems via Simulation, in *Handbook of Simulation,* J. Banks, ed., John Wiley, New York (1998).

Goldsman, D., and L.W. Schruben: Asymptotic Properties of Some Confidence Interval Estimators for Simulation Output, *Management Sci., 30:* 1217–1225 (1984).

Goldsman, D., and L.W. Schruben: New Confidence Interval Estimators Using Standardized Time Series, *Management Sci., 36:* 393–397 (1990).

Goldsman, D., L.W. Schruben, and J.J. Swain: Tests for Transient Means in Simulated Times Series, *Naval Research Logistics, 41:* 171–187 (1994).

Gonzalez, T., S. Sahni, and W.R. Franta: An Efficient Algorithm for the Kolmogorov-Smirnov and Lilliefors Tests, *Assoc. Comput. Mach. Trans. Math. Software, 3:* 60–64 (1977).

Grassman, W.K.: Rethinking the Initialization Bias Problem in Steady-State Discrete Event Simulation, *Proc. 2011 Winter Simulation Conference,* Phoenix, Arizona, pp. 593–599 (2011).

Gray, D., and D. Goldsman: Indifference-Zone Selection Procedures for Choosing the Best Airspace Configuration, *Proc. 1988 Winter Simulation Conference,* San Diego, pp. 445–450 (1988).

Grigoryev, I., and A. Borshchev: *AnyLogic 6 in Three Days: A Quick Course in Simulation Modeling,* AnyLogic North America, Lisle, Illinois (2012).

Grosenbaugh, L.R.: More on Fortran Random Number Generators, *Commun. Assoc. Comput. Mach., 12*: 639 (1969).

Gross, D., J.F. Shortle, J.M. Thompson, and C.M. Harris: *Fundamentals of Queueing Theory,* 4th ed., John Wiley, New York (2009).

Gupta, S.S.: On a Decision Rule for a Problem in Ranking Means, Mimeograph Series No. 150, Institute of Statistics, University of North Carolina, Chapel Hill (1956).

Gupta, S.S.: On Some Multiple Decision (Selection and Ranking) Rules, *Technometrics, 7:* 225–245 (1965).

Gupta, S.S., and T.J. Santner: On Selection and Ranking Procedures—A Restricted Subset Selection Rule, *Proc. 39th Session of the International Statistical Institute,* Vienna, Vol. 1 (1973).

Gupta, U.G.: Using Citation Analysis to Explore the Intellectual Base, Knowledge Dissemination, and Research Impact of *Interfaces* (1970–1992), *Interfaces, 27:* 85–101 (1997).

Gürkan, G., A.Y. Özge, and S.M. Robinson: Sample-Path Solution of Stochastic Variational Inequalities, *Math. Programming, 84:* 313–333 (1999).

Haberman, S.J.: A Warning on the Use of Chi-Squared Statistics with Frequency Tables with Small Expected Cell Counts, *J. Am. Statist. Assoc., 83:* 555–560 (1988).

Hahn, G.J, and S.S. Shapiro: *Statistical Models in Engineering,* John Wiley, New York (1994).

Haider, S.W., D.G. Noller, and T.B. Robey: Experiences with Analytic and Simulation Modeling for a Factory of the Future Project at IBM, *Proc. 1986 Winter Simulation Conference,* Washington, D.C., pp. 641–648 (1986).

Halton, J.H.: A Retrospective and Prospective Survey of the Monte Carlo Method, *SIAM Rev., 12:* 1–63 (1970).

Hammersley, J.M., and D.C. Handscomb: *Monte Carlo Methods,* Methuen, London (1964).

Hammersley, J.M., and K.W. Morton: A New Monte Carlo Technique: Antithetic Variates, *Proc. Camb. Phil. Soc., 52:* 449–475 (1956).

Harrell, C.R., B.K. Ghosh, and R.O. Bowden: *Simulation Using ProModel,* 3d ed. McGraw-Hill, New York (2012).

Haykin, S.O.: *Neural Networks and Learning Machines*, 3d ed., Prentice Hall, Upper Saddle River, New Jersey (2009).

Hazra, M.M., D.J. Morrice, and S.K. Park: A Simulation Clock-Based Solution to the Frequency Domain Experiment Indexing Problem, *IIE Trans., 29:* 769–782 (1997).

He, D., S.E. Chick, and C.-H. Chen: The Opportunity Cost and OCBA Selection Procedures in Ordinal Optimization, *IEEE Trans. on Systems, Man, and Cybernetics—Part C, 37:* 951–961 (2007).

Healey, C., S. Andradóttir, and S.-H. Kim: Efficient Comparison of Constrained Systems Using Dormancy, *Eur. J. Operational Research, 224:* 340–352 (2013).

Heath, B.L., R.R. Hill, and F. Ciarello: A Survey of Agent-Based Modeling Practices (January 1998 to July 2008), *J. of Artificial Societies and Social Simulation, 12*, http://jasss.soc.surrey.ac.uk/12/4/9.html (2009).

Heathcote, C.R., and P. Winer: An Approximation to the Moments of Waiting Times, *Operations Res., 17:* 175–186 (1969).

Heidelberger, P.: Variance Reduction Techniques for the Simulation of Markov Processes, I: Multiple Estimates, *IBM J. Res. Develop., 24:* 570–581 (1980).

Heidelberger, P.: Fast Simulation of Rare Events in Queueing and Reliability Models, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 5:* 43–85 (1995).

Heidelberger, P., and D.L. Iglehart: Comparing Stochastic Systems Using Regenerative Simulation with Common Random Numbers, *Adv. Appl. Prob., 11:* 804–819 (1979).

Heidelberger, P., and P.A.W. Lewis: Quantile Estimation in Dependent Sequences, *Operations Res., 32:* 185–209 (1984).

Heidelberger, P., and P.D. Welch: Adaptive Spectral Methods for Simulation Output Analysis, *IBM J. Res. Develop., 25:* 860–876 (1981a).

Heidelberger, P., and P.D. Welch: A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations, *Commun. Assoc. Comput. Mach., 24:* 233–245 (1981b).

Heidelberger, P., and P.D. Welch: Simulation Run Length Control in the Presence of an Initial Transient, *Operations Res., 31:* 1109–1144 (1983).

Heikes, R.G., D.C. Montgomery, and R.L. Rardin: Using Common Random Numbers in Simulation Experiments—An Approach to Statistical Analysis, *Simulation, 25:* 81–85 (1976).

Henderson, D., S.H. Jacobson, and A.W. Johnson: The Theory and Practice of Simulated Annealing, in *Handbook of Metaheuristics,* F. Glover and G. Kochenberger, eds., Springer, New York (2003).

Henderson, S.G.: Input Model Uncertainty: Why Do We Care and What Should We Do about It?, *Proc. 2003 Winter Simulation Conference,* New Orleans, pp. 90–100 (2003).

Henderson, S.G., and P.W. Glynn: Regenerative Steady-State Simulation of Discrete-Event Systems, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 11:* 313–345 (2001).

Henriksen, J.O.: An Improved Events List Algorithm, *Proc. 1977 Winter Simulation Conference,* San Diego, California, pp. 547–557 (1977).

Henriksen, J.O.: Event List Management—A Tutorial, *Proc. 1983 Winter Simulation Conference,* Washington, D.C., pp. 543–551 (1983).

Hernandez, A.S., T.W. Lucas, and M. Carlyle: Constructing Nearly Orthogonal Latin Hypercubes for Any Nonsaturated Run-Variable Combination, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 22,* No. 4, Article 20 (2012).

Hesterberg, T.C., and B.L. Nelson: Control Variates for Probability and Quantile Estimation, *Management Sci., 44:* 1295–1312 (1998).

Hill, R.R., L.E. Champagne, and J.C. Price: Using Agent-based Simulation and Game Theory to Examine the WWII Bay of Biscayne U-Boat Campaign, *J. of Defense Modeling and Simulation, 1*: 99–109 (2004).

Hill, R.R., and C.H. Reilly: Composition for Multivariate Random Variables, *Proc. 1994 Winter Simulation Conference,* Orlando, pp. 332–339 (1994).

Ho, Y.C., and X.R. Cao: *Discrete Event Dynamic Systems and Perturbation Analysis,* Kluwer Academic Publishers, Amsterdam (1991).

Ho, Y.C., C.G. Cassandras, C.H. Chen, and L.Y. Dai: Ordinal Optimization and Simulation, *J. Operational Res. Soc., 51:* 490–500 (2000).

Ho, Y.C., R. Sreenivas, and P. Vakili: Ordinal Optimization of Discrete Event Dynamic Systems, *Discrete Event Dynamic Systems: Theory and Applications, 2:* 61–88 (1992).

Hoad, K.A., S. Robinson, and R. Davies: Automating Warm-Up Length Estimation, *J. Operational Res. Soc., 61:* 1389–1403 (2009).

Hoaglin, D.C., F. Mosteller, and J.W. Tukey: *Understanding Robust and Exploratory Data Analysis,* John Wiley, New York (1983).

Hochberg, Y., and A.C. Tamhane: *Multiple Comparison Procedures,* John Wiley, New York (1987).

Hogg, R.V., and A.F. Craig: *Introduction to Mathematical Statistics,* 5th ed., Prentice-Hall, Upper Saddle River, New Jersey (1995).

Holland, J.H.: *Hidden Order: How Adaptation Builds Complexity*, Basic Books, New York (1995).

Hong, L.J., and B.L. Nelson: The Tradeoff between Sampling and Switching: New Sequential

Procedures for Indifference-Zone Selection, *IIE Trans., 37:* 623–634 (2005).
Hong, L.J., and B.L. Nelson: Discrete Optimization via Simulation Using COMPASS, *Operations Res., 54:* 115–129 (2006).
Hood, S.J., and P.D. Welch: Experimental Design Issues in Simulation with Examples from Semiconductor Manufacturing, *Proc. 1992 Winter Simulation Conference,* Washington, D.C., pp. 255–263 (1992).
Hood, S.J., and P.D. Welch: Response Surface Methodology and Its Application in Simulation, *Proc. 1993 Winter Simulation Conference,* Los Angeles, pp. 115–122 (1993).
Hopp, W.J., and M.L. Spearman: *Factory Physics*, 3d ed., Waveland Press, Long Grove, Illinois (2011).
Hörmann, W.: A Rejection Technique for Sampling from T-Concave Distributions, *Assoc. Comput. Mach. Trans. Math. Software, 21:* 182–193 (1995).
Hörmann, W., and G. Derflinger: Rejection-Inversion to Generate Variates from Monotone Discrete Distributions, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 6:* 169–184 (1996).
Hörmann, W., and J. Leydold: Continuous Random Variate Generation by Fast Numerical Inversion, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 13:* 347–362 (2003).
Hörmann, W., J. Leydold, and G. Derflinger: *Automatic Nonuniform Random Variate Generation,* Springer-Verlag, New York (2004).
Hsu, D.A., and J.S. Hunter: Analysis of Simulation-Generated Responses Using Autoregressive Models, *Management Sci., 24:* 181–190 (1977).
Hsu, J.C.: Constrained Two-Sided Simultaneous Confidence Intervals for Multiple Comparisons with the Best, *Ann. Statist., 12:* 1136–1144 (1984).
Hsu, J.C.: *Multiple Comparisons: Theory and Methods,* Chapman & Hall, London, England (1996).
Hsu, J.C., and B.L. Nelson: Control Variates for Quantile Estimation, *Management Sci., 36:* 835–851 (1990).
Hu, J., M.C. Fu, and S.I. Marcus: A Model Reference Adaptive Search Method for Global Optimization, *Operations Res., 55:* 549–568 (2007).
Hull, T.E., and A.R. Dobell: Random Number Generators, *SIAM Rev., 4:* 230–254 (1962).
Hunter, S.R., and R. Pasupathy: Optimal Sampling Laws for Stochastically Constrained Simulation Optimization on Finite Sets, to appear in *INFORMS J. Comput.* (2013).
Hussey, J.R., R.H. Myers, and E.C. Houck: Correlated Simulation Experiments in First-Order Response Surface Designs, *Operations Res., 35:* 744–758 (1987).
Hutchinson, D.W.: A New Uniform Pseudorandom Number Generator, *Commun. Assoc. Comput. Mach., 9:* 432–433 (1966).
Iglehart, D.L.: Simulating Stable Stochastic Systems, V: Comparison of Ratio Estimators, *Naval Res. Logist. Quart., 22:* 553–565 (1975).
Iglehart, D.L., and P.W. Lewis: Regenerative Simulation with Internal Controls, *J. Assoc. Comput. Mach., 26:* 271–282 (1979).
Ilachinski, A.: *Artificial War: Multiagent-Based Simulation of Combat,* World Scientific Publishing Company, Singapore (2004).
Imagine That, Inc.: ExtendSim simulation software, www.extendsim.com (2013).
INCONTROL Simulation Solutions, Enterprise Dynamics simulation software, www.incontrolsim.com (2013).
Irish, T.H., D.C. Dietz, and K.W. Bauer, Jr.: Replicative Use of an External Analytical Model in Simulation Variance Reduction, *IIE Trans., 35:* 879–894 (2003).
Jacobson, S.H., A.H. Buss, and L.W. Schruben: Driving Frequency Selection for Frequency Domain Simulation Experiments, *Operations Res., 39:* 917–924 (1991).
Jain, R., and I. Chlamtac: The P^2 Algorithm for Dynamic Calculation of Quantiles and Histograms without Storing Observations, *Commun. Assoc. Comput. Mach., 28:* 1076–1085 (1985).
Jefferson, D.R.: Virtual Time, *Assoc. Comput. Mach. Trans. Programming Languages and Systems, 7:* 404–425 (1985).
Joanes, D.N., and C.A. Gill: Comparing Measures of Sample Skewness and Kurtosis, *The Statistician, 47:* 183–189 (1998).
Jöhnk, M.D.: Erzeugung von Betaverteilten und Gammaverteilten Zufallszahlen, *Metrika, 8:* 5–15 (1964).

Johnson, M.A., S. Lee, and J.R. Wilson: Experimental Evaluation of a Procedure for Estimating Nonhomogeneous Poisson Processes Having Cyclic Behavior, *ORSA J. Comput., 6:* 356–368 (1994a).

Johnson, M.A., S. Lee, and J.R. Wilson: NPPMLE and NPPSIM: Software for Estimating and Simulating Nonhomogeneous Poisson Processes Having Cyclic Behavior, *Operations Res. Letters, 15:* 273–282 (1994b).

Johnson, M.E.: *Multivariate Statistical Simulation,* John Wiley, New York (1987).

Johnson, M.E., and J.S. Ramberg: Transformations of the Multivariate Normal Distribution with Applications to Simulation, Los Alamos Sci. Lab. Tech. Rep. LA-UR-77-2595, Los Alamos, New Mexico (1978).

Johnson, M.E., C. Wang, and J.S. Ramberg: Generation of Continuous Multivariate Distributions for Statistical Applications, *Am. J. Math. and Management Sci., 4:* 96–119 (1984).

Johnson, N.L., S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions,* Volume 1, 2d ed., Houghton Mifflin, Boston (1994).

Johnson, N.L., S. Kotz, and N. Balakrishnan, *Continuous Univariate Distributions,* Volume 2, 2d ed., Houghton Mifflin, Boston (1995).

Johnson, N.L., S. Kotz, and N. Balakrishnan: *Discrete Multivariate Distributions,* John Wiley, New York (1997).

Johnson, N.L., S. Kotz, and A.W. Kemp: *Univariate Discrete Distributions*, 2d ed., Houghton Mifflin, Boston (1992).

Joines, J.A., and S.D. Roberts: Object-Oriented Simulation, in *Handbook of Simulation*, J. Banks, ed., John Wiley, New York (1998).

Jones, D.W.: An Empirical Comparison of Priority-Queue and Event-Set Implementations, *Commun. Assoc. Comput. Mach., 29:* 300–311 (1986).

Jones, R.M., and K.S. Miller: On the Multivariate Lognormal Distribution, *J. Indust. Math., 16:* 63–76 (1966).

Kabirian, A., and S. Ólafsson: Continuous Optimization via Simulation Using Golden Region Search, *Eur. J. Operational Res., 208:* 19–27 (2011).

Kachitvichyanukul, V., and B.W. Schmeiser: Binomial Random Variate Generation, *Commun. Assoc. Comput. Mach., 31:* 216–222 (1988).

Kallenberg, W.C.M., J. Oosterhoff, and B.F. Schriever: The Number of Classes in Chi-Squared Goodness-of-Fit Tests, *J. Am. Statist. Assoc., 80:* 959–968 (1985).

Kaminsky, F.C., and D.L. Rumpf: Simulating Nonstationary Poisson Processes: A Comparison of Alternatives Including the Correct Approach, *Simulation, 29:* 17–20 (1977).

Kang, K., K.H. Doerr, and S.M. Sanchez: A Design of Experiments Approach to Readiness Risk Analysis, *Proc. 2006 Winter Simulation Conference,* Monterey, California, pp. 1332–1339 (2006).

Kao, E.P.C., and S.L. Chang: Modeling Time-Dependent Arrivals to Service Systems: A Case in Using a Piecewise-Polynomial Rate Function in a Nonhomogeneous Poisson Process, *Management Sci., 34:* 1367–1379 (1988).

Kapuściński, R., and S. Tayur: A Capacitated Production-Inventory Model with Periodic Demand, *Operations Res., 46:* 899–911 (1998).

Keefer, D.L., and S.E. Bodily: Three-Point Approximations for Continuous Random Variables, *Management Sci., 29:* 595–609 (1983).

Kelton, W.D.: The Startup Problem in Discrete-Event Simulation, Technical Report 80-1, Dept. of Industrial Engineering, University of Wisconsin, Madison, Wisconsin (1980).

Kelton, W.D.: Transient Exponential-Erlang Queues and Steady-State Simulation, *Commun. Assoc. Comput. Mach., 28:* 741–749 (1985).

Kelton, W.D.: Random Initialization Methods in Simulation, *IIE Trans., 21:* 355–367 (1989).

Kelton, W.D., and A.M. Law: A New Approach for Dealing with the Startup Problem in Discrete Event Simulation, *Naval Res. Logist. Quart., 30:* 641–658 (1983).

Kelton, W.D., and A.M. Law: The Transient Behavior of the *M/M/s* Queue, with Implications for Steady-State Simulation, *Operations Res., 33:* 378–396 (1985).

Kelton, W.D., R.P. Sadowski, and N.B. Swets: *Simulation with Arena,* 5th ed., McGraw-Hill, New York (2010).

Kelton, W.D., J.S. Smith, D.T. Sturrock: *Simio & Simulation: Modeling, Analysis, Applications*, 2d ed., McGraw-Hill Learning Solutions, Indianapolis, Indiana (2011).

Kendall, M.G., and B. Babington-Smith: Randomness and Random Sampling Numbers, *J. Roy. Statist. Soc., 101A:* 147–166 (1938).

Kendall, M.G., and A. Stuart: *The Advanced Theory of Statistics,* Vol. 2, 4th ed., Griffin, London (1979).

Kendall, M.G., A. Stuart, and J.K. Ord: *The Advanced Theory of Statistics,* Vol. 1, 5th ed., Oxford University Press, New York (1987).

Kennedy, W.J., Jr., and J.E. Gentle: *Statistical Computing,* Marcel Dekker, New York (1980).

Kernighan, B.W., and D.M. Ritchie: *The C Programming Language,* 2d ed., Prentice-Hall, Englewood Cliffs, New Jersey (1988).

Kim, S., R. Pasupathy, and S.G. Henderson: A Guide to Sample-Average Approximation, in *Handbook on Simulation Optimization*, M.C. Fu, ed., Springer, New York (2013).

Kim, S.-H., and B.L. Nelson: A Fully Sequential Procedure for Indifference-Zone Selection in Simulation, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 11:* 251–273 (2001).

Kim, S.-H., and B.L. Nelson: Selecting the Best System, in *Elsevier Handbooks in Operations Research and Management Science,* H.G. Henderson and B.L. Nelson, eds., Elsevier, New York (2006a).

Kim, S.-H., and B.L. Nelson: On the Asymptotic Validity of Fully Sequential Selection Procedures for Steady-State Simulation, *Operations Res., 54:* 475–488 (2006b).

Kim, S.-H., and B.L. Nelson: Recent Advances in Ranking and Selection, *Proc. 2007 Winter Simulation Conference*, Washington, D.C., pp. 162–172 (2007).

Kinderman, A.J., and J.F. Monahan: Computer Generation of Random Variables Using the Ratio of Uniform Deviates, *Assoc. Comput. Mach. Trans. Math. Software, 3:* 257–260 (1977).

Kinderman, A.J., and J.G. Ramage: Computer Generation of Normal Random Variables, *J. Am. Statist. Assoc., 71:* 893–896 (1976).

Kleijnen, J.P.C.: *Statistical Techniques in Simulation,* Pt. I, Marcel Dekker, New York (1974).

Kleijnen, J.P.C.: Analysis of Simulation with Common Random Numbers: A Note on Heikes et al. (1976), *Simuletter, 11:* 7–13 (1979).

Kleijnen, J.P.C.: Regression Metamodels for Simulation with Common Random Numbers: Comparison of Validation Tests and Confidence Intervals, *Management Sci., 38:* 1164–1185 (1992).

Kleijnen, J.P.C.: Experimental Design for Sensitivity Analysis, Optimization, and Validation of Simulation Models, in *Handbook of Simulation,* J. Banks, ed., John Wiley, New York (1998).

Kleijnen, J.P.C.: *Design and Analysis of Simulation Experiments*, Springer, New York (2007).

Kleijnen, J.P.C.: Design of Experiments: Overview, *Proc. 2008 Winter Simulation Conference,* Miami, Florida, pp. 479–488 (2008).

Kleijnen, J.P.C.: Kriging Metamodeling in Simulation: A Review, *Eur. J. Operational Res., 192:* 707–716 (2009).

Kleijnen, J.P.C., B. Bettonvil, and F. Persson: Screening for the Important Factors in Large Discrete-Event Simulation Models: Sequential Bifurcation and Its Applications, in *Screening: Methods for Experimentation in Industry, Drug Discovery, and Genetics,* A.M. Dean and S.M. Lewis, eds., Springer, New York (2006).

Kleijnen, J.P.C., B. Bettonvil, and W. Van Groenendaal: Validation of Trace-Driven Simulation Models: A Novel Regression Test, *Management Sci., 44:* 812–819 (1998).

Kleijnen, J.P.C., R.C.H. Cheng, and B. Bettonvil: Validation of Trace-Driven Simulation Models: More on Bootstrap Techniques, *Proc. 2000 Winter Simulation Conference,* Orlando, pp. 882–892 (2000).

Kleijnen, J.P.C., R.C.H. Cheng, and B. Bettonvil: Validation of Trace-Driven Simulation Models: Bootstrap Tests, *Management Sci., 47:* 1533–1538 (2001).

Kleijnen, J.P.C., and D. Deflandre: Validation of Regression Metamodels in Simulation: Bootstrap Approach, *Eur. J. Operational Res., 170:* 120–131 (2006).

Kleijnen, J.P.C., S.M. Sanchez, T.W. Lucas, and T.M. Cioppa: A User's Guide to the Brave New World of Designing Simulation Experiments, *INFORMS J. Comput., 17:* 263–289 (2005).

Kleijnen, J.P.C., and R.G. Sargent: A Methodology for Fitting and Validating Metamodels in Simulation, *Eur. J. Operational Res., 120:* 14–29 (2000).

Klein, R.W., and S.D. Roberts: A Time-Varying Poisson Arrival Process Generator, *Simulation, 43*: 193–195 (1984).

Knepell, P.L., and D.C. Arangno: *Simulation Validation: A Confidence Assessment Methodology,* IEEE Computer Society Press, Los Alamitos, California (1993).

Knuth, D.E.: *The Art of Computer Programming, Vol. 1: Fundamental Algorithms,* 3d ed., Addison-Wesley, Reading, Massachusetts (1997).

Knuth, D.E.: *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms,* 3d ed., Addison-Wesley, Reading, Massachusetts (1998a).

Knuth, D.E.: *The Art of Computer Programming, Vol. 3: Sorting and Searching,* 2d ed., Addison-Wesley, Reading, Massachusetts (1998b).

Koenig, L.W., and A.M. Law: A Procedure for Selecting a Subset of Size *m* Containing the *l* Best of *k* Independent Normal Populations, with Applications to Simulation, Tech. Rep. 82-9, Department of Management Information Systems, University of Arizona, Tucson (1982).

Koenig, L.W., and A.M. Law: A Procedure for Selecting a Subset of Size *m* Containing the *l* Best of *k* Independent Normal Populations, *Commun. Statist.—Simulation and Computation, 14:* 719–734 (1985).

Kotz, S., N. Balakrishnan, and N.L. Johnson: *Continuous Multivariate Distributions, Vol. 1, Models and Applications,* 2d ed., Wiley, New York (2000).

Kronmal, R.A., and A.V. Peterson, Jr.: On the Alias Method for Generating Random Variables from a Discrete Distribution, *Am. Statistician, 33:* 214–218 (1979).

Kronmal, R.A., and A.V. Peterson, Jr.: A Variant of the Acceptance-Rejection Method for Computer Generation of Random Variables, *J. Am. Statist. Assoc., 76:* 446–451 (1981).

Kronmal, R.A., and A.V. Peterson, Jr.: Corrigenda, *J. Am. Statist. Assoc., 77:* 954 (1982).

Kuhl, M.E., H. Damerdji, and J.R. Wilson: Estimating and Simulating Poisson Processes with Trends or Asymmetric Cyclic Effects, *Proc. 1997 Winter Simulation Conference,* Atlanta, pp. 287–295 (1997).

Kuhl, M.E., S.G. Sumant, and J.R. Wilson: An Automated Multiresolution Procedure for Modeling Complex Arrival Processes, *INFORMS J. Comput., 18:* 3–18 (2006).

Kuhl, F.S., R.M. Weatherly, and J.S. Dahmann: *Creating Computer Simulation Systems: An Introduction to the High Level Architecture,* Prentice-Hall, Upper Saddle River, New Jersey (2000).

Kuhl, M.E., and J.M. Wilson: Least Squares Estimation of Nonhomogeneous Poisson Processes, *J. Statist. Comput. Simul., 67:* 75–108 (2000).

Kuhl, M.E., and J.M. Wilson: Modeling and Simulating Poisson Processes Having Trends or Nontrigonometric Cyclic Effects, *Eur. J. Operational Res., 133:* 566–582 (2001).

Kuhl, M.E., J.R. Wilson, and M.A. Johnson: Estimating and Simulating Poisson Processes Having Trends or Multiple Periodicities, *IIE Trans., 29:* 201–211 (1997).

Kumar, S., and D.A. Nottestad: Capacity Design: An Application Using Discrete-Event Simulation and Designed Experiments, *IIE Trans., 38:* 729–736 (2006).

Kwon, C., and J.D. Tew: Strategies for Combining Antithetic Variates and Control Variates in Designed Simulation Experiments, *Management Sci., 40:* 1021–1034 (1994).

Lada, E.K., N.M. Steiger, and J.R. Wilson: Performance Evaluation of Recent Procedures for Steady-State Simulation Analysis, *IIE Trans., 38:* 711–727 (2006).

Lada, E.K., N.M. Steiger, and J.R. Wilson: SBatch: A Spaced Batch Means Procedure for Steady-State Simulation Analysis, *J. of Simulation, 2:* 170–185 (2008).

Lada, E.K., and J.R. Wilson: A Wavelet-Based Spectral Procedure for Steady-State Simulation Analysis, *Eur. J. Operational Res., 174:* 1769–1801 (2006a).

Lada, E.K., and J.R. Wilson: Performance Evaluation of Spectral Procedures for Simulation Analysis, *Proc. 2006 Winter Simulation Conference*, Monterey, California, pp. 198–207 (2006b).

Lada, E.K., J.R. Wilson, N.M. Steiger, and J.A. Joines: Performance of a Wavelet-Based Spectral Procedure for Steady-State Simulation Analysis, *INFORMS J. Comput., 19:* 150–160 (2007).

Laguna, M.: Scatter Search, in the *Handbook of Applied Optimization,* pp. 183–193, P.M. Pardalos and M.G.C. Resende, eds., Oxford University Press, New York (2002).

Laguna, M.: OptQuest: Optimization of Complex Systems, www.opttek.com, OptTek Systems, Inc., Boulder, Colorado (2011).

Laguna, M., and R. Marti: *Scatter Search: Methodology and Implementations in C,* Kluwer Academic Publishers, Boston (2003).

Lanchester, F.W.: *Aircraft in Warfare: the Dawn of the Fourth Arm*, Constable and Company Ltd., London (1916).

Lane, M.S., A.H. Mansour, and J.L. Harpell: Operations Research Techniques: A Longitudinal Update 1973–1988, *Interfaces, 23:* 63–68 (1993).

Lanner Group Ltd.: WITNESS simulation software, www.lanner.com (2013).

Lavenberg, S.S., T.L. Moeller, and C.H. Sauer: Concomitant Control Variables Applied to the Regenerative Simulation of Queueing Systems, *Operations Res., 27:* 134–160 (1979).

Lavenberg, S.S., T.L. Moeller, and P.D. Welch: Statistical Results on Control Variables with Application to Queueing Network Simulation, *Operations Res., 30:* 182–202 (1982).

Lavenberg, S.S., and P.D. Welch: Using Conditional Expectation to Reduce Variance in Discrete-Event Simulation, *Proc. 1979 Winter Simulation Conference,* San Diego, pp. 291–294 (1979).

Lavenberg, S.S., and P.D. Welch: A Perspective on the Use of Control Variables to Increase the Efficiency of Monte Carlo Simulations, *Management Sci., 27:* 322–335 (1981).

Law, A.M.: Efficient Estimators for Simulated Queueing Systems, *Univ. California Operations Res. Center ORC 74-7,* Berkeley (1974).

Law, A.M.: Efficient Estimators for Simulated Queueing Systems, *Management Sci., 22:* 30–41 (1975).

Law, A.M.: Confidence Intervals in Discrete-Event Simulation: A Comparison of Replication and Batch Means, *Naval Res. Logist. Quart., 24:* 667–678 (1977).

Law, A.M.: Statistical Analysis of the Output Data from Terminating Simulations, *Naval Res. Logist. Quart., 27:* 131–143 (1980).

Law, A.M.: Statistical Analysis of Simulation Output Data, *Operations Res., 31:* 983–1029 (1983).

Law, A.M.: Simulation Model's Level of Detail Determines Effectiveness, *Ind. Eng., 23:* 16, 18 (October 1991).

Law, A.M.: How to Conduct a Successful Simulation Study, *Proc. 2003 Winter Simulation Conference,* New Orleans, pp. 66–70 (2003).

Law, A.M.: How to Build Valid and Credible Simulation Models, *Proc. 2009 Winter Simulation Conference,* Austin, Texas, pp. 24–33 (2009).

Law, A.M.: A Practitioner's Perspective on Simulation Validation, VV&A Recommended Practices Guide, Modeling & Simulation Coordination Office, http://vva.msco.mil, Alexandria, Virginia (2011).

Law, A.M., and J.S. Carson: A Sequential Procedure for Determining the Length of a Steady-State Simulation, *Operations Res., 27:* 1011–1025 (1979).

Law, A.M., and W.D. Kelton: Confidence Intervals for Steady-State Simulations, II: A Survey of Sequential Procedures, *Management Sci., 28:* 550–562 (1982).

Law, A.M., and W.D. Kelton: Confidence Intervals for Steady-State Simulations, I: A Survey of Fixed Sample Size Procedures, *Operations Res., 32:* 1221–1239 (1984).

Law, A.M., W.D. Kelton, and L.W. Koenig: Relative Width Sequential Confidence Intervals for the Mean, *Commun. Statist., B10:* 29–39 (1981).

Law, A.M., and M.G. McComas: How Simulation Pays Off, *Manuf. Eng., 100:* 37–39 (February 1988).

Law, A.M., and M.G. McComas: Pitfalls to Avoid in the Simulation of Manufacturing Systems, *Ind. Eng., 21:* 28–31 (May 1989).

Lawless, J.F.: Statistical Models and Methods for Lifetime Data, 2d ed., John Wiley, New York (2003).

L'Ecuyer, P.: Efficient and Portable Combined Random Number Generators, *Commun. Assoc. Comput. Mach., 31:* 742–749 and 774 (1988).

L'Ecuyer, P.: Efficiency Improvement and Variance Reduction, *Proc. 1994 Winter Simulation Conference,* Orlando, pp. 122–132 (1994a).

L'Ecuyer, P.: Uniform Random Number Generation, *Annals of Operations Res., 53:* 77–120 (1994b).

L'Ecuyer, P.: Combined Multiple Recursive Random Number Generators, *Operations Res., 44:* 816–822 (1996a).

L'Ecuyer, P.: Maximally Equidistributed Combined Tausworthe Generators, *Math. of Computation, 65:* 203–213 (1996b).

L'Ecuyer, P.: Random Number Generation, in *Handbook of Simulation,* J. Banks, ed., John Wiley, New York (1998).

L'Ecuyer, P.: Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators, *Operations Res., 47:* 159–164 (1999a).

L'Ecuyer, P.: Tables of Maximally Equidistributed Combined LFSR Generators, *Math. of Computation, 68:* 261–269 (1999b).

L'Ecuyer, P.: Uniform Random Number Generation, in *Handbooks in Operations Research and Management Science: Simulation,* S.G. Henderson and B.L. Nelson, eds., Elsevier, New York (2006).

L'Ecuyer, P.: Random Number Generation, in *Handbook of Computational Statistics,* 2d ed., J.E. Gentle, W. Härdle, and Y. Mori, eds., Springer, New York (2012).

L'Ecuyer, P., F. Blouin, and R. Couture: A Search for Good Multiple Recursive Random Number Generators, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 17:* 98–111 (1993).

L'Ecuyer, P., J.-F. Cordeau, and R. Simard: Close-Point Spatial Tests and Their Application to Random Number Generators, *Operations Res., 48:* 308–317 (2000).

L'Ecuyer, P., and F. Panneton: F_2 Linear Random Number Generators, in *Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman,* C. Alexopoulos, D. Goldsman, and J.R. Wilson, eds., Springer, New York (2009).

L'Ecuyer, P., and R. Simard: On the Performance of Birthday Spacings Tests with Certain Families of Random Number Generators, *Math. and Comp. in Simul., 55:* 131–137 (2001).

L'Ecuyer, P., and R. Simard: TEST01: A C Library for Empirical Testing of Random-Number Generators, *Assoc. Comput. Mach. Trans. Math. Software, 33,* No. 4, Article 22 (2007).

L'Ecuyer, P., R. Simard, E.J. Chen, and W.D. Kelton: An Object-Oriented Random-Number Package with Many Long Streams and Substreams, *Operations Res., 50:* 1073–1075 (2002).

L'Ecuyer, P., R. Simard, and S. Wegenkittl: Sparse Serial Tests of Uniformity for Random Number Generators, *SIAM J. Scientific Computing, 24:* 652–668 (2002).

L'Ecuyer, P., and S. Tezuka: Structural Properties for Two Classes of Combined Random Number Generators, *Math. of Computation, 57:* 735–746 (1991).

Lee, L.H., N.A. Pujowidianto, L.-W. Li, C.-H. Chen, and C.M. Yap: Approximate Simulation Budget Allocation for Selecting the Best System in the Presence of Stochastic Constraints, *IEEE Trans. on Auto. Cont., 57:* 2940–2945 (2012).

Lee, S., J.R. Wilson, and M.M. Crawford: Modeling and Simulation of a Nonhomogeneous Poisson Process Having Cyclic Behavior, *Commun. Statist., B20:* 777–809 (1991).

Leemis, L.: Nonparametric Estimation of the Cumulative Intensity Function for a Nonhomogeneous Poisson Process, *Management Sci., 37:* 886–900 (1991).

Leemis, L.: Building Credible Input Models, *Proc. 2004 Winter Simulation Conference,* Washington, D.C., pp. 29–40 (2004).

Lehmer, D.H.: Mathematical Methods in Large-Scale Computing Units, *Ann. Comput. Lab. Harvard Univ., 26:* 141–146 (1951).

Levasseur, J.A.: The Case for Object-Oriented Simulation, *OR/MS Today, 23:* 65–67 (August 1996).

Levin, B., and J. Reeds: Compound Multinomial Likelihood Functions Are Unimodal: Proof of a Conjecture of I.J. Good, *Ann. Statist., 5:* 79–87 (1977).

Lew, J.: Making City Planning a Game, *The New York Times* (June 15, 1989).

Lewis, P.A.W.: Generating Negatively Correlated Gamma Variates Using the Beta-Gamma Transformation, *Proc. 1983 Winter Simulation Conference,* Washington, D.C., pp. 175–176 (1983).

Lewis, P.A.W., A.S. Goodman, and J.M. Miller: A Pseudorandom-Number Generator for the System/360, *IBM Syst. J., 8:* 136–146 (1969).

Lewis, P.A.W., E. McKenzie, and D.K. Hugus: Gamma Processes, *Comm. Statist. Stoch. Models, 5:* 1–30 (1989).

Lewis, P.A.W., and G.S. Shedler: Statistical Analysis of Non-Stationary Series of Events in a Data Base System, *IBM J. Res. Dev., 20:* 465–482 (1976).

Lewis, P.A.W., and G.S. Shedler: Simulation of Nonhomogeneous Poisson Process by Thinning, *Nav. Res. Logist. Quart., 26:* 403–413 (1979).

Lewis, T.G., and W.H. Payne: Generalized Feedback Shift Register Pseudorandom-Number Algorithm, *J. Assoc. Comput. Mach., 20:* 456–468 (1973).

Leydold, J.: Automatic Sampling with the Ratio-of-Uniforms Method, *Assoc. Comput. Mach. Trans. Math. Software, 26:* 78–98 (2000).

Lilliefors, H.W.: On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown, *J. Am. Statist. Assoc., 62:* 399–402 (1967).

Lilliefors, H.W.: On the Kolmogorov-Smirnov Test for the Exponential Distribution with Mean Unknown, *J. Am. Statist. Assoc., 64:* 387–389 (1969).

Littell, R.C., J.T. McClave, and W.W. Offen: Goodness-of-Fit Tests for the Two Parameter Weibull Distribution, *Commun. Statist., B8:* 257–269 (1979).

Livny, M., B. Melamed, and A.K. Tsiolis: The Impact of Autocorrelation on Queueing Systems, *Management Sci, 39:* 322–339 (1993).

Loeppky, J.L., J. Sacks, and W.J. Welch: Choosing the Sample Size of a Computer Experiment: A Practical Guide, *Technometrics, 51:* 366–376 (2009).

Lophaven, S.N., H.B. Nielsen, and J. Sondergaard: DACE: A MATLAB Kriging Toolbox, Version 2.0, IMM Technical University of Denmark, Lyngby, Denmark (2002).

Lotka, A.J.: *Elements of Physical Biology,* Williams and Wilkins, Philadelphia, Pennsylvania (1925).

Macal, C.M.: Agent Based Modeling and Artificial Life, *Encyclopedia of Complexity and Systems Science,* pp. 112–131, Springer, New York (2009).

Macal, C.M., and M.J. North: Agent-Based Modeling and Simulation: ABMS Examples, *Proc. 2008 Winter Simulation Conference,* Miami, Florida, pp. 101–112 (2008).

Macal, C.M., and M.J. North: Tutorial on Agent-Based Modelling and Simulation, *J. of Simulation, 4:*151–162 (2010).

Macal, C.M., and M.J. North: Introductory Tutorial: Agent-Based Modeling and Simulation, *Proc. 2011 Winter Simulation Conference,* Phoenix, Arizona, pp. 1456–1469 (2011).

Macal, C.M., and M.J. North: Personal Communication (2013).

Macal, C.M., M.J. North, and D.A. Samuelson: Agent-Based Simulation, in *Encyclopedia of Operations Research and Management Science,* 3d ed., S.I. Gass and M.C. Fu, eds., Springer, New York (2013).

MacKay, N.: Lancaster Combat Models, *Mathematics Today, 42:* 170–173 (2006).

MacLaren, M.D., and G. Marsaglia: Uniform Random Number Generators, *J. Assoc. Comput. Mach., 12:* 83–89 (1965).

MacLaren, M.D., G. Marsaglia, and T.A. Bray: A Fast Procedure for Generating Exponential Random Variables, *Commun. Assoc. Comput. Mach., 7:* 298–300 (1964).

Margolin, B.H., and W. Maurer: Tests of the Kolmogorov-Smirnov Type for Exponential Data with Unknown Scale, and Related Problems, *Biometrika, 63:* 149–160 (1976).

Marsaglia, G.: Generating Exponential Random Variables, *Ann. Math. Statist., 32:* 899–902 (1961).

Marsaglia, G.: Generating Discrete Random Variables in a Computer, *Commun. Assoc. Comput. Mach., 6*: 37–38 (1963).

Marsaglia, G.: Random Numbers Fall Mainly in the Planes, *Natl. Acad. Sci. Proc., 61:* 25–28 (1968).

Marsaglia, G.: The Exact-Approximation Method for Generating Random Variables in a Computer, *J. Am. Statist. Assoc., 79:* 218–22 (1984).

Marsaglia, G.: The Marsaglia Random Number CD, including the DIEHARD Battery of Tests of Randomness, Department of Statistics, Florida State University, Tallahassee, Florida (1995). See http://stat.fsu.edu/pub/diehard.

Marsaglia, G., and T.A. Bray: A Convenient Method for Generating Normal Variables, *SIAM Rev., 6:* 260–264 (1964).

Marsaglia, G., and T.A. Bray: One-Line Random Number Generators and Their Use in Combinations, *Commun. Assoc. Comput. Mach., 11:* 757–759 (1968).

Marse, K., and S.D. Roberts: Implementing a Portable FORTRAN Uniform (0,1) Generator, *Simulation, 41:* 135–139 (1983).

Marshall, A.W., and I. Olkin: A Multivariate Exponential Distribution, *J. Am. Statist. Assoc., 62:* 30–44 (1967).

MASON: http://cs.gmu.edu/~eclab/projects/mason (2013).
MathWorks: MATLAB software, www.matlab.com (2013).
Matsumoto, M., and Y. Kurita: Twisted GFSR Generators, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 2:* 179–194 (1992).
Matsumoto, M., and Y. Kurita: Twisted GFSR Generators II, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 4:* 254–266 (1994).
Matsumoto, M., and Y. Kurita: Strong Deviations from Randomness in *m*-Sequences Based on Trinomials, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 6:* 99–106 (1996).
Matsumoto, M., and T. Nishimura: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 8:* 3–30 (1998).
McCormack, W.M., and R.G. Sargent: Analysis of Future Event Set Algorithms for Discrete Event Simulation, *Commun. Assoc. Comput. Mach., 24:* 801–812 (1981).
McIntosh, G.C., D.P. Galligan, M.A. Anderson, and M.K. Lauren, MANA (Map Aware Non-Uniform Automata), Version 4 Users Manual, Defence Technology Agency, New Zealand Defence Force, Auckland, New Zealand (2007).
McLean, C., and F. Riddick: The IMS Mission Architecture for Distributed Manufacturing Simulation, *Proc. 2000 Winter Simulation Conference,* Orlando, pp. 1539–1548 (2000).
McLeod, I.: A Remark on AS 183. An Efficient and Portable Pseudo-random Number Generator, *Appl. Statist., 34:* 198–200 (1985).
Meketon, M.S., and B.W. Schmeiser: Overlapping Batch Means: Something for Nothing?, *Proc. 1984 Winter Simulation Conference,* Dallas, pp. 227–230 (1984).
Menon, M.V.: Estimation of the Shape and Scale Parameters of the Weibull Distribution, *Technometrics, 5:* 175–182 (1963).
Meterelliyoz, M., C. Alexopoulos, and D. Goldsman: Folded Overlapping Variance Estimators for Simulation, *Eur. J. Operational Research, 220:* 135–146 (2012).
Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs,* 3d ed., Springer, New York (2013).
Miller, J.O., B.L. Nelson, and C.H. Reilly: Efficient Multinomial Selection in Simulation, *Naval Res. Logist., 45:* 459–482 (1998).
Miller, R.G., Jr.: The Jackknife—A Review, *Biometrika, 61:* 1–15 (1974).
Milton, R.C., and R. Hotchkiss: Computer Evaluation of the Normal and Inverse Normal Distribution Functions, *Technometrics, 11:* 817–822 (1969).
Minar, N., R. Burkhart, C. Langton, and M. Askenazi: The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations, Working Paper 96-06-042, Santa Fe Institute, Santa Fe, New Mexico (1996).
Minh, D.L.: A Variant of the Conditional Expectation Variance Reduction Technique and Its Application to the Simulation of the *GI/G/*1 Queues, *Management Sci., 35:* 1334–1340 (1989).
Minitab Inc.: Minitab statistical software, www.minitab.com (2013).
Mitchell, B.: Variance Reduction by Antithetic Variates in *GI/G/*1 Queueing Simulations, *Operations Res., 21:* 988–997 (1973).
Mitchell, C.R., and A.S. Paulson: *M/M/*1 Queues with Interdependent Arrival and Service Processes, *Nav. Res. Logist. Quart., 26:* 47–56 (1979).
Mitchell, C.R., A.S. Paulson, and C.A. Beswick: The Effect of Correlated Exponential Service Times on Single Server Tandem Queues, *Nav. Res. Logist. Quart., 24:* 95–112 (1977).
Miyatake, O., M. Ichimura, Y. Yoshizawa, and H. Inoue: Mathematical Analysis of Random-Number Generator Using Gamma Rays I, *Math. Jap., 28:* 399–414 (1983).
Modeling & Simulation Coordination Office: VV&A Recommended Practices Guide, http://vva.msco.mil, Alexandria, Virginia (2011).
Mokashi, A.C., J.J. Tejada, S. Yousefi, T. Xu, J.R. Wilson, A. Tafazzoli, and N.M. Steiger: Performance Comparison of MSER-5 and N-Skart on the Simulation Start-Up Problem, *Proc. 2010 Winter Simulation Conference,* Baltimore, Maryland, pp. 971–982 (2010).
Montgomery, D.C.: *Design and Analysis of Experiments*, 8th ed., John Wiley, New York (2013).
Montgomery, D.C., and G.C. Runger: *Applied Statistics and Probability for Engineers,* 5th ed., John Wiley, New York (2011).
Mood, A.M., F.A. Graybill, and D.C. Boes: *Introduction to the Theory of Statistics,* 3d ed.,

McGraw-Hill, New York (1974).
Morgan, B.J.T.: *Elements of Simulation,* Chapman & Hall, London (1984).
Moro, B.: The Full Monte, *Risk, 8:* 57–58 (1995).
Morrice, D.J., and I.R. Bardhan: A Weighted Least-Squares Approach to Computer Simulation Factor Screening, *Operations Res., 43:* 792–806 (1995).
Morrice, D.J., and L.W. Schruben: Simulation Factor Screening Using Cross-Spectral Methods, *Operations Res. Letters, 13:* 247–257 (1993a).
Morrice, D.J., and L.W. Schruben: Simulation Factor Screening Using Harmonic Analysis, *Management Sci., 39:* 1459–1476 (1993b).
Murphy, W.S., Jr., and M.A. Flourney: Simulation Crisis Communications, *Proc. 2002 Winter Simulation Conference,* San Diego, pp. 954–959 (2002).
Murray, J.R.: Stochastic Initialization in Steady-State Simulations, Ph.D. Dissertation, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor (1988).
Murray, J.R., and W.D. Kelton: The Transient Behavior of the $M/E_k/2$ Queue and Steady-State Simulation, *Computers and Operations Res., 15:* 357–367 (1988).
Myers, R.H., D.C. Montgomery, and C.M. Anderson-Cook: *Response Surface Methodology: Process and Product Optimization Using Designed Experiments,* 3d ed., John Wiley, New York (2009).
Nakayama, M.K.: Fast Simulation Methods for Highly Dependable Systems, *Proc. 1994 Winter Simulation Conference,* Orlando, pp. 221–228 (1994a).
Nakayama, M.K.: A Characterization of the Simple Failure Biasing Method for Simulations of Highly Reliable Markovian Systems, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 4:* 52–88 (1994b).
Nakayama, M.K.: Multiple-Comparison Procedures for Steady-State Simulations, *Ann. Statist., 25:* 2433–2450 (1997).
Nakayama, M.K.: Multiple Comparisons with the Best Using Common Random Numbers, *J. Statist. Plan. Inf., 85:* 37–48 (2000).
Nance, R.E., and C. Overstreet, Jr.: Implementation of Fortran Random-Number Generators on Computers with One's Complement Arithmetic, *J. Statist. Comput. Simul., 4:* 235–243 (1975).
Nance, R.E., and C. Overstreet, Jr.: Some Experimental Observations on the Behavior of Composite Random-Number Generators, *Operations Res., 26:* 915–935 (1978).
Nance, R.E., and R.G. Sargent: Perspectives on the Evolution of Simulation, *Operations Res., 50:* 161–172 (2002).
Naylor, T.H.: *Computer Simulation Experiments with Models of Economic Systems,* John Wiley, New York (1971).
Naylor, T.H., and J.M. Finger: Verification of Computer Simulation Models, *Management Sci., 14:* 92–101 (1967).
Nelson, B.L.: A Decomposition Approach to Variance Reduction, *Proc. 1985 Winter Simulation Conference,* San Francisco, pp. 23–33 (1985).
Nelson, B.L.: Decomposition of Some Well-Known Variance Reduction Techniques, *J. Statist. Comput. Simul., 23:* 183–209 (1986).
Nelson, B.L.: A Perspective on Variance Reduction in Dynamic Simulation Experiments, *Commun. Statist., B16:* 385–426 (1987a).
Nelson, B.L.: Some Properties of Simulation Interval Estimators under Dependence Induction, *Operations Res. Letters, 6:* 169–176 (1987b).
Nelson, B.L.: Variance Reduction for Simulation Practitioners, *Proc. 1987 Winter Simulation Conference,* Atlanta, pp. 43–51 (1987c).
Nelson, B.L.: Batch Size Effects on the Efficiency of Control Variates in Simulation, *Eur. J. Operational Res., 43:* 184–196 (1989).
Nelson, B.L.: Control-Variate Remedies, *Operations Res., 38:* 974–992 (1990a).
Nelson, B.L.: Variance Reduction in the Presence of Initial-Condition Bias, *IIE Trans., 22:* 340–350 (1990b).
Nelson, B.L.: Robust Multiple Comparisons under Common Random Numbers, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 3:* 225–243 (1993).

Nelson, B.L.: The More Plot: Displaying Measures of Risk & Error from Simulation Output, *Proc. 2008 Winter Simulation Conference*, Miami, Florida, pp. 413–416 (2008).

Nelson, B.L., and J.C. Hsu: Control-Variate Models of Common Random Numbers for Multiple Comparisons with the Best, *Management Sci., 39*: 989–1001 (1993).

Nelson, B.L., and F.J. Matejcik: Using Common Random Numbers for Indifference-Zone Selection and Multiple Comparisons in Simulation, *Management Sci., 41:* 1935–1945 (1995).

Nelson, B.L., B.W. Schmeiser, M.R. Taaffe, and J. Wang: Approximation-Assisted Point Estimation, *Operations Res. Letters, 20:* 109–118 (1997).

Nelson, B.L, J. Swann, D. Goldsman, and W.-M.T. Song: Simple Procedures for Selecting the Best System When the Number of Alternatives Is Large, *Operations Res., 49:* 950–963 (2001).

Nelson, B.L., and M. Yamnitsky: Input Modeling Tools for Complex Problems, *Proc. 1998 Winter Simulation Conference,* Washington, D.C., pp. 105–112 (1998).

NetLogo: http://ccl.northwestern.edu/netlogo (2013).

Nicola, V.F., P. Shahabuddin, and M.K. Nakayama: Techniques for Fast Simulation of Models of Highly Dependable Systems, *IEEE Trans. Reliability, 50:* 246–264 (2001).

Niederreiter, H.: Quasi-Monte Carlo Methods and Pseudo-random Numbers, *Bull. Am. Math. Soc., 84:* 957–1041 (1978).

Nilsson, F., and V. Darley: On Complex Adaptive Systems and Agent-Based Modelling for Improved Decision-Making in Manufacturing and Logistics Settings: Experiences from a Packaging Company, *International J. of Operations & Production Management, 26:* 1351–1373 (2006).

North, M.J., N.T. Collier, J. Ozik, E.R. Tatara, C.M. Macal, M. Bragen, and P. Sydelko: Complex Adaptive Systems Modeling with Repast Simphony, *Complex Adaptive Systems Modeling, 1:* 1–26 (2013).

North., M.J., and C.M. Macal: *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*, Oxford University Press, New York (2007).

Nozari, A., S.F. Arnold, and C.D. Pegden: Statistical Analysis with the Schruben and Margolin Correlation Induction Strategy, *Operations Res., 35:* 127–139 (1987).

Ólaffson, S., and J. Kim: Simulation Optimization, *Proc. 2002 Winter Simulation Conference*, San Diego, pp. 79–84 (2002).

Owen, D.B.: *Handbook of Statistical Tables*, Addison-Wesley, Reading, Massachusetts (1962).

Pace, D.K.: Thoughts about the Conceptual Model, *Proc. of the Simulation Interoperability Workshop,* Orlando, Florida, Paper Number 009 (Spring 2003).

Page, E.S.: On Monte Carlo Methods in Congestion Problems: II. Simulation of Queueing Systems, *Operations Res., 13:* 300–305 (1965).

Panneton, F., P. L'Ecuyer, and M. Matsumoto: Improved Long-Period Generators Based on Linear Recurrences Modulo 2, *Assoc. Comput. Mach. Trans. Math. Software, 32:* 1–16 (2006).

Parker, J., and J.M. Epstein: A Distributed Platform for Global-Scale Agent-Based Models of Disease Propagation, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 22,* No. 1, Article 2 (2011).

Pasupathy, R., and B.W. Schmeiser: The Initial Transient in Steady-State Point Estimation: Contexts, a Bibliography, the MSE Criterion, and the MSER Statistic, *Proc. 2010 Winter Simulation Conference*, Baltimore, Maryland, pp. 184–197 (2010).

Pawlikowski, K.: Steady-State Estimation of Queueing Processes: A Survey of Problems and Solutions, *Assoc. Comput. Mach Computing Surveys, 22:* 123–170 (1990).

Payne, W.H., J.R. Rabung, and T.P. Bogyo: Coding the Lehmer Pseudorandom Number Generator, *Commun. Assoc. Comput. Mach., 12:* 85–86 (1969).

Pearson, K.: On a Criterion That a Given System of Deviations from the Probable in the Case of a Correlated System of Variables Is Such That It Can Be Reasonably Supposed to Have Arisen in Random Sampling, *Phil. Mag.* (5), *50:* 157–175 (1900).

Pegden, C.D.: Personal Communication (2010).

Peng, Y., C.-H. Chen, M.C. Fu, and J.Q. Hu: Efficient Simulation Resource Sharing and Allocation for Selecting the Best, *IEEE Trans. on Auto. Cont., 58:* 1017–1023 (2013).

Peterson, A.V., Jr., and R.A. Kronmal: On Mixture Methods for the Computer Generation of Random Variables, *Am. Statistician, 36:* 184–191 (1982).

Peterson, A.V., Jr., and R.A. Kronmal: Analytic Comparison of Three General-Purpose Methods for the Computer Generation of Discrete Random Variables, *Appl. Statist., 32:* 276–286 (1983).

Pettitt, A.N., and M.A. Stephens: The Kolmogorov-Smirnov Goodness-of-Fit Statistic with Discrete and Grouped Data, *Technometrics, 19:* 205–210 (1977).

Pichitlamken, J., B.L. Nelson, and L.J. Hong: A Sequential Procedure for Neighborhood Selection-of-the-Best in Optimization via Simulation, *Eur. J. Operational Res., 173:* 283–298 (2006).

Porcaro, D.: Simulation Modeling and DOE, *Industrial Engineering Solutions:* 24–30 (September 1996).

Poropudas, J., and K. Virtanen: Simulation Metamodeling with Dynamic Bayesian Networks, *Eur. J. Operational Res., 214:* 644–655 (2011).

Porta Nova, A.M., and J.R. Wilson: Selecting Control Variates to Estimate Multiresponse Simulation Metamodels, *Eur. J. Operational Res., 71:* 80–94 (1993).

Posadas, S., and E.P. Paulo: Stochastic Simulation of a Commander's Decision Cycle, *Military Operations Res., 8:* 21–43 (2003).

Press, W.H., S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery: *Numerical Recipes: The Art of Scientific Computing,* 3d ed., Cambridge University Press, Cambridge, England (2007).

Pritsker, A.A.B.: *Introduction to Simulation and SLAM II,* 4th ed., John Wiley, New York (1995).

ProcessModel, Inc.: ProcessModel simulation software, www.processmodel.com (2013).

ProModel Corporation: MedModel simulation software, www.promodel.com (2013).

ProModel Corporation: Process Simulator simulation software, www.promodel.com (2013).

ProModel Corporation: ProModel simulation software, www.promodel.com (2013).

ProModel Corporation: ServiceModel simulation software, www.promodel.com (2013).

Pujowidianto, N.A., L.H. Lee, C.-H. Chen, and C.M. Yap: Optimal Budget Allocation for Constrained Optimization, *Proc. 2009 Winter Simulation Conference,* Austin, Texas, pp. 584–589 (2009).

Raatikainen, K.E.E.: Sequential Procedure for Simultaneous Estimation of Several Percentiles, *Trans. of the Society for Comp. Simulation, 7:* 21–24 (1990).

Railsback, S.F., and V. Grimm: *Agent-Based and Individual-Based Modeling: A Practical Introduction,* Princeton University Press, Princeton, New Jersey (2012).

Rajasekaran, S., and K.W. Ross: Fast Algorithms for Generating Discrete Random Variates with Changing Distributions, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 3:* 1–19 (1993).

Ramberg, J.S., and P.R. Tadikamalla: On the Generation of Subsets of Order Statistics, *J. Statist. Comput. Simul., 6:* 239–241 (1978).

Rand Corporation: *A Million Random Digits with 100,000 Normal Deviates,* Free Press, Glencoe, Illinois (1955).

Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavior Model, *Proc. of the ACM 14th Annual Conference on Computer Graphics and Interactive Techniques*, Anaheim, California (1987).

Rinott, Y.: On Two-Stage Selection Procedures and Related Probability-Inequalities, *Commun. Statist., A7:* 799–811 (1978).

Ripley, B.D.: *Stochastic Simulation,* John Wiley, New York (1987).

Riverbed Technologies: OPNET Modeler simulation software, www.opnet.com (2013).

Robbins, H., G. Simons, and N. Starr: A Sequential Analogue of the Behrens-Fisher Problem, *Ann. Math. Statist., 38:* 1384–1391 (1967).

Robinson, S.: *Simulation: The Practice of Model Development and Use,* Wiley, Chichester, United Kingdom (2004).

Robinson, S.: Conceptual Modelling for Simulation Part I: Definition and Requirements, *J. Operational Res. Soc., 59:* 291–304 (2008a).

Robinson, S.: Conceptual Modelling for Simulation Part II: A Framework for Conceptual Modeling, *J. Operational Res. Soc., 59:* 278–290 (2008b).

Robinson, S.M.: Analysis of Sample-Path Optimization, *Math. of Operations Res., 21:* 513–528 (1996).

Rockwell Automation: Arena simulation software, www.arenasimulation.com (2013).

Rogue Wave Software: IMSL Numerical Libraries, www.roguewave.com (2013).

Rönngren, R., and R. Ayani: A Comparative Study of Parallel and Sequential Priority Queue Algorithms, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 7:* 157–209 (1997).
Ronning, G.: A Simple Scheme for Generating Multivariate Gamma Distributions with Nonnegative Covariance Matrix, *Technometrics, 19:* 179–183 (1977).
Ross, S.M.: *Introduction to Probability Models,* 8th ed., Academic Press, San Diego (2003).
Rubinstein, R.Y.: *Simulation and the Monte Carlo Method,* John Wiley, New York (1981).
Rubinstein, R.Y.: *Monte Carlo Optimization, Simulation and Sensitivity of Queueing Networks,* Krieger Publishing Co., Malabar, Florida (1992).
Rubinstein, R.Y., and D.P. Kroese: *Simulation and the Monte Carlo Method*, 2d ed., John Wiley, New York (2008).
Rubinstein, R.Y., and R. Marcus: Efficiency of Multivariate Control Variates in Monte Carlo Simulation, *Operations Res., 33:* 661–667 (1985).
Rubinstein, R.Y., B. Melamed, and A. Shapiro: *Modern Simulation and Modeling,* John Wiley, New York (1998).
Rubinstein, R.Y., G. Samorodnitsky, and M. Shaked: Antithetic Variates, Multivariate Dependence, and Simulation of Complex Stochastic Systems, *Management Sci., 31:* 66–77 (1985).
Rubinstein, R.Y., and A. Shapiro: *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Method,* John Wiley, New York (1993).
Rudolph, E., and D.M. Hawkins: Random-Number Generators in Cyclic Queuing Applications, *J. Statist. Comput. Simul., 5:* 65–71 (1976).
Rukhin, A., J. Soto, J. Nechvatel, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800–22, National Institute for Standards and Technology, Gaithersburg, Maryland (2001). See http://csrs.nist.gov/rng/.
Runciman, N., N. Vagenas, and T. Corkal: Simulation of Haulage Truck Loading Techniques in an Underground Mine Using WITNESS, *Simulation, 68:* 291–299 (1997).
Russell, E.C.: *Simulation and SIMSCRIPT II.5,* CACI, Inc., Los Angeles (1976).
Sacks, J., W.J. Welch, T.J. Mitchell, and H.P. Wynn: Design and Analysis of Computer Simulation Experiments, *Statistical Science, 4:* 409–423 (1989).
Samuelson, D.A: Mass Egress and Post-Disaster, *OR/MS Today:* 20–24 (October 2007a).
Samuelson, D.A.: Panel: Agent Based Modeling of Mass Egress and Evacuations, *Proc. of the 2007 Winter Simulation Conference,* Washington, D.C., pp. 1247–1251 (2007b).
Sanchez, P.J., and S.M. Sanchez: Design of Frequency Domain Experiments for Discrete-Valued Factors, *Appl. Math. and Computation, 42:* 1–21 (1991).
Sanchez, P.J., and L.W. Schruben: Simulation Factor Screening Using Frequency Domain Methods: An Illustrative Example, Working Paper 87–013, Systems and Industrial Engineering Dept., University of Arizona, Tucson (1987).
Sanchez, P.J., and K.P. White: Interval Estimation Using Replication/Deletion and MSER Truncation, *Proc. 2011 Winter Simulation Conference,* Phoenix, Arizona, pp. 488–494 (2011).
Sanchez, S.M., T.W. Lucas, P.J. Sanchez, C.J. Nannini, and H. Wan: Designs for Large-Scale Simulation Experiments, with Applications to Defense and Homeland Security, in *Design and Analysis of Experiments: Special Designs and Experiments,* K. Hinkelmann, ed., John Wiley, New York (2012).
Sanchez, S.M., and P.J. Sanchez: Very Large Fractional Factorial and Central Composite Designs, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul.,15:* 362–377 (2005).
Sanchez, S.M., and H. Wan: Work Smarter, Not Harder: A Tutorial on Designing and Conducting Simulation Experiments, *Proc. 2012 Winter Simulation Conference,* Berlin, Germany (2012).
Sanchez, S.M., H. Wan, and T.W. Lucas: Two-phase Screening Procedures for Simulation Experiments, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 19,* No. 2, Article 7 (2009).
Santner, T.J.: A Restricted Subset Selection Approach to Ranking and Selection Problems, *Ann. Statist., 3:* 334–349 (1975).
Santner, T.J., B.J. Williams, and W.I. Notz: *The Design and Analysis of Computer Experiments,* Springer, New York (2003).
Sargent, R.G.: Event Graph Modelling for Simulation with an Application to Flexible Manufac-

turing Systems, *Management Sci., 34:* 1231–1251 (1988).
Sargent, R.G.: Verification and Validation of Simulation Models, *J. of Simulation, 6:* 1–13 (2012).
Sargent, R.G., K. Kang, and D. Goldsman: An Investigation of Finite-Sample Behavior of Confidence Interval Estimators, *Operations Res., 40:* 898–913 (1992).
Sargent, R.G., and T.K. Som: Current Issues in Frequency Domain Experimentation, *Management Sci., 38:* 667–687 (1992).
SAS Institute Inc.: JMP statistical software, www.jmp.com (2013).
SCALABLE Network Technologies, Inc.: QualNet simulation software, www.scalable-networks.com (2013).
Schelling, T.C.: Dynamic Models of Segregation, *J. of Mathematical Sociology:* 143–186 (1971).
Scheuer, E.M., and D.S. Stoller: On the Generation of Normal Random Vectors, *Technometrics, 4:* 278–281 (1962).
Schmeiser, B.W.: Generation of the Maximum (Minimum) Value in Digital Computer Simulation, *J. Statist. Comput. Simul., 8:* 103–115 (1978a).
Schmeiser, B.W.: The Generation of Order Statistics in Digital Computer Simulation: A Survey, *Proc. 1978 Winter Simulation Conference,* Miami, pp. 137–140 (1978b).
Schmeiser, B.W.: Generation of Variates from Distribution Tails, *Operations Res., 28:* 1012–1017 (1980a).
Schmeiser, B.W.: Random Variate Generation: A Survey, *Proc. 1980 Winter Simulation Conference*, Orlando, pp. 79–104 (1980b).
Schmeiser, B.W.: Batch Size Effects in the Analysis of Simulation Output, *Operations Res., 30:* 556–568 (1982).
Schmeiser, B.W., and A.J.G. Babu: Beta Variate Generation via Exponential Majorizing Functions, *Operations Res., 28:* 917–926 (1980).
Schmeiser, B.W., and V. Kachitvichyanukul: Poisson Random Variate Generation, School of Industrial Engineering Research Memorandum 81-4, Purdue Univ., West Lafayette, Indiana (1981).
Schmeiser, B.W., and V. Kachitvichyanukal: Non-Inverse Correlation Induction: Guidelines for Algorithm Development, *J. Comput. and Applied Math., 31:* 173–180 (1990).
Schmeiser, B.W., and R. Lal: Squeeze Methods for Generating Gamma Variates, *J. Am. Statist. Assoc., 75:* 679–382 (1980).
Schmeiser, B.W., and R. Lal: Bivariate Gamma Random Vectors, *Operations Res., 30:* 355–374 (1982).
Schmeiser, B.W., and M.A. Shalaby: Acceptance/Rejection Methods for Beta Variate Generation, *J. Am. Statist. Assoc., 75:* 673–678 (1980).
Schmeiser, B.W., and W.T. Song: Batching Methods in Simulation Output Analysis: What We Know and What We Don't, *Proc. 1996 Winter Simulation Conference,* San Diego, 122–127 (1996).
Schmeiser, B.W., and M.R. Taaffe: Time-Dependent Queueing Network Approximations as Simulation External Control Variates, *Operations Res. Letters, 16:* 1–9 (1994).
Schmidt, J.W., and R.E. Taylor: *Simulation and Analysis of Industrial Systems,* Richard D. Irwin, Homewood, Illinois (1970).
Schrage, L.: A More Portable Random-Number Generator, *Assoc. Comput. Mach. Trans. Math. Software, 5:* 132–138 (1979).
Schriber, T.J.: *Simulation Using GPSS,* John Wiley, New York (1974).
Schriber, T.J., and R.W. Andrews: An ARMA-Based Confidence Interval for the Analysis of Simulation Output, *Am. J. Math. Management Sci., 4:* 345–373 (1984).
Schruben, L.W.: Designing Correlation Induction Strategies for Simulation Experiments, in *Current Issues in Computer Simulation,* N.R. Adam and A. Dogramici, eds., Academic Press, New York (1979).
Schruben, L.W.: Establishing the Credibility of Simulations, *Simulation, 34:* 101–105 (1980).
Schruben, L.W.: Detecting Initialization Bias in Simulation Output, *Operations Res., 30:* 569–590 (1982).
Schruben, L.W.: Confidence Interval Estimation Using Standardized Time Series, *Operations Res., 31:* 1090–1108 (1983a).
Schruben, L.W.: Simulation Modeling with Event Graphs, *Commun. Assoc. Comput. Mach., 26:*

957–963 (1983b).
Schruben, L.W., and V.J. Cogliano: An Experimental Procedure for Simulation Response Surface Model Identification, *Commun. Assoc. Comput. Mach., 30:* 716–730 (1987).
Schruben, L.W., and B.H. Margolin: Pseudorandom Number Assignment in Statistically Designed Simulation and Distribution Sampling Experiments, *J. Am. Statist. Assoc., 73:* 504–520 (1978).
Schruben, L.W., T.M. Roeder, W.K. Chan, P. Hyden, and M. Freimer: Advanced Event Scheduling Methodology, *Proc. 2003 Winter Simulation Conference,* New Orleans, pp. 159–165 (2003).
Schruben, L.W., H. Singh, and L. Tierney: Optimal Tests for Initialization Bias in Simulation Output, *Operations Res., 31:* 1167–1178 (1983).
Schucany, W.R.: Order Statistics in Simulation, *J. Statist. Comput. Simul., 1:* 281–286 (1972).
Schwefel, H.-P.: *Evolution and Optimum Seeking,* John Wiley, New York (1995).
Scott, D.W.: On Optimal and Data-Based Histograms, *Biometrika, 66:* 605–610 (1979).
SEAS: U.S. Air Force Space Command, Space and Missile Systems Center, www.teamseas.com, Los Angeles, California (2013).
Seila, A.F.: Spreadsheet Simulation, *Proc. 2005 Winter Simulation Conference,* Orlando, Florida, pp. 33–40 (2005).
Seila, A.F., V. Ceric, and P. Tadikamalla: *Applied Simulation Modeling,* Brooks-Cole, Belmont, California (2003).
Shahabuddin, P.: Importance Sampling for the Simulation of Highly Reliable Markovian Systems, *Management Sci., 40:* 333–352 (1994).
Shahabuddin, P.: Rare Event Simulation in Stochastic Models, *Proc. 1995 Winter Simulation Conference,* Washington, D.C., pp. 178–185 (1995).
Shannon, R.E.: *Systems Simulation: The Art and Science,* Prentice-Hall, Englewood Cliffs, New Jersey (1975).
Shanthikumar, J.G.: Discrete Random Variate Generation Using Uniformization, *Eur. J. Operational Res., 21:* 387–398 (1985).
Shanthikumar, J.G., and J.A. Buzacott: *Stochastic Models of Manufacturing Systems,* Prentice-Hall, Englewood Cliffs, New Jersey (1993).
Shapiro, S.S., and M.B. Wilk: An Analysis of Variance Test for Normality (Complete Samples), *Biometrika, 52:* 591–611 (1965).
Shedler, G.S.: *Regenerative Stochastic Simulation,* Academic Press, San Diego (1993).
Shen, H., and H. Wan: Controlled Sequential Factorial Design for Simulation Factor Screening, *Eur. J. Operational Res., 198:* 511–519 (2009).
Shen, H., H. Wan, and S.M. Sanchez: A Hybrid Method for Simulation Factor Screening, *Naval Res. Logistics, 57:* 45–57 (2010).
Shi, L., and S. Ólafsson: Nested Partitioned Method for Global Optimization, *Operations Res., 48:* 390–407 (2000).
Siebel, F., and L. Kellam: The Virtual World of Agent-Based Modeling: Procter & Gamble's Dynamic Supply Chain, *Perspectives on Business Innovation, 9:* 22–27 (2003).
Siemens: Plant Simulation simulation software, http://www.plm.automation.siemens.com/en_us/products/tecnomatix/free-trial/plant-simulation.shtml (2013).
Sigal, C.E., A.A.B. Pritsker, and J.J. Solberg: The Use of Cutsets in Monte Carlo Analysis of Stochastic Networks, *Math. Comput. Simul., 21:* 376–384 (1979).
Silverman, B.W.: *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, London (1986).
Simio LLC: Simio simulation software, www.simio.com (2013).
SIMUL8 Corporation: SIMUL8 simulation software, www.simul8.com (2013).
Slifker, J., and S.S. Shapiro: On the Johnson System of Distributions, *Technometrics, 22:* 239–246 (1980).
Smith, R.L.: Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed over Bounded Regions, *Operations Res., 32:* 1296–1308 (1984).
Snell, M., and L.W. Schruben: Weighting Simulation Data to Reduce Initialization Effects, Technical Report 395, School of Industrial Engineering, Cornell University, Ithaca, New York (1979).
Som, T.K., and R.G. Sargent: A Formal Development of Event Graphs as an Aid to Structured and Efficient Simulation Programs, *ORSA J. Comput., 1:* 107–125 (1989).

Song, W.-M.T., and B.W. Schmeiser: Optimal Mean-Squared-Error Batch Sizes, *Management Sci., 41:* 110–123 (1995).

Spall, J.C.: *Introduction to Stochastic Search and Optimization,* John Wiley, New York (2003).

Spieckermann, S., K. Gutenschwager, H. Heinzel, and S. Voss: Simulation-based Optimization in the Automotive Industry—A Case Study on Body Shop Design, *Simulation, 75:* 276–286 (2000).

Srikant, R., and W. Whitt: Variance Reduction in Simulations of Loss Models, *Operations Res., 47:* 509–523 (1999).

Stadlober, E.: The Ratio of Uniforms Approach for Generating Discrete Random Variates, *J. Comput. and Applied Math., 31:* 181–189 (1990).

Stanfield, P.M., J.R. Wilson, G.A. Mirka, N.F. Glasscock, J.P. Psihogios, and J.R. Davis: Multivariate Input Modeling with Johnson Distributions, *Proc. 1996 Winter Simulation Conference,* San Diego, pp. 1457–1464 (1996).

Stat-Ease, Inc.: Design-Expert statistical software, www.stat-ease.com (2013).

Staum, J.: Better Simulation Metamodeling: The Why, What, and How of Stochastic Kriging, *Proc. 2009 Winter Simulation Conference*, Austin, Texas, pp. 119–133 (2009).

Stefánescu, S., and I. Váduva: On Computer Generation of Random Vectors by Transformation of Uniformly Distributed Vectors, *Computing, 39:* 141–153 (1987).

Steiger, N.M., E.K. Lada, J.R. Wilson, J.A. Joines, C. Alexopoulos, and D. Goldsman: ASAP3: A Batch Means Procedure for Steady-State Simulation Analysis, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 15:* 39–73 (2005).

Steiger, N.M., and J.R. Wilson: Convergence Properties of the Batch Means Method for Simulation Output Analysis, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 13:* 277–293 (2001).

Steiger, N.M., and J.R. Wilson: An Improved Batch Means Procedure for Simulation Output Analysis, *Management Sci., 48:* 1569–1586 (2002).

Stephens, M.A.: EDF Statistics for Goodness of Fit and Some Comparisons, *J. Am. Statist. Assoc., 69:* 730–737 (1974).

Stephens, M.A.: Asymptotic Results for Goodness-of-Fit Statistics with Unknown Parameters, *Ann. Statist., 4:* 357–369 (1976).

Stephens, M.A.: Goodness of Fit for the Extreme Value Distribution, *Biometrika, 64:* 583–588 (1977).

Stephens, M.A.: Tests of Fit for the Logistic Distribution Based on the Empirical Distribution Function, *Biometrika, 66:* 591–595 (1979).

Sterman, J.D.: *Business Dynamics: Systems Thinking and Modeling for a Complex World,* McGraw-Hill, New York (2001).

Stidham, S: A Last Word on $L = \lambda w$, *Operations Res., 22:* 417–421 (1974).

Sullivan, D.W., and J.R. Wilson: Restricted Subset Selection Procedures for Simulation, *Operations Res., 37:* 52–71 (1989).

Swain, J.J., S. Venkatraman, and J.R. Wilson: Least-Squares Estimation of Distribution Functions in Johnson's Translation System, *J. Statist. Comput. Simul., 29:* 271–297 (1988).

Swart, W., and L. Donno: Simulation Modeling Improves Operations, Planning, and Productivity for Fast Food Restaurants, *Interfaces, 11:6:* 35–47 (1981).

Swisher, J.R., P.D. Hyden, S.H. Jacobson, and L.W. Schruben: A Survey of Recent Advances in Discrete Input Parameter Discrete-Event Simulation Optimization, *IIE Trans., 36:* 591–600 (2004).

Swisher, J.R., S.H. Jacobson, and E. Yücesan: Discrete-Event Simulation Optimization Using Ranking, Selection, and Multiple Comparison Procedures: A Survey, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 13:* 134–154 (2003).

Tadikamalla, P.R.: Computer Generation of Gamma Random Variables-II, *Commun. Assoc. Comput. Mach., 21:* 925–928 (1978).

Tadikamalla, P.R.: Kolmogorov-Smirnov Type Test-Statistics for the Gamma, Erlang-2, and the Inverse Gaussian Distributions When the Parameters Are Unknown, *Commun. Statist.-Simulation and Computation, 19:* 305–314 (1990).

Tadikamalla, P.R., and M.E. Johnson: A Complete Guide to Gamma Variate Generation, *Am. J. Math. Management Sci., 1:* 78–95 (1981).

Tafazzoli, A., N.M. Steiger, and J.R. Wilson: N-Skart: A Nonsequential Skewness- and Autoregression-Adjusted Batch-Means Procedure for Simulation Analysis, *IEEE Trans. Auto. Cont., 56:* 254–264 (2011).

Tafazzoli, A., and J.R. Wilson: Skart: A Skewness- and Autoregression-Adjusted Batch-Means Procedure for Simulation Analysis, *IIE Trans., 43:* 110–128 (2011).

Tafazzoli, A., J.R. Wilson, E.K. Lada, and N.M. Steiger: Performance of Skart: A Skewness- and Autoregression-Adjusted Batch Means Procedure for Simulation Analysis, *INFORMS J. Comput., 23:* 297–314 (2011).

Tang, W.T., R.S.M. Goh, and I.L.-J. Thng: Ladder Queue: An $O(1)$ Priority Queue Structure for Large-Scale Discrete Event Simulation, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 15:* 175–204 (2005).

Tausworthe, R.C.: Random Numbers Generated by Linear Recurrence Modulo Two, *Math. Comput., 19:* 201–209 (1965).

Taylor, J.G.: *Lanchester Models of Warfare*, INFORMS, Hanover, Maryland (1983).

Tekin, E., and I. Sabuncuoglu: Simulation Optimization: A Comprehensive Review on Theory and Applications, *IIE Trans., 36:* 1067–1081 (2004).

Tew, J.D., and J.R. Wilson: Validation of Simulation Analysis Methods for the Schruben-Margolin Correlation-Induction Strategy, *Operations Res., 40:* 87–103 (1992).

Tezuka, S.: *Uniform Random Numbers: Theory and Practice,* Kluwer Academic Publishers, Norwell, Massachusetts (1995).

Thoman, D.R., L.J. Bain, and C.E. Antle: Inferences on the Parameters of the Weibull Distribution, *Technometrics, 11:* 445–460 (1969).

Thomson, W.E.: ERNIE—A Mathematical and Statistical Analysis, *J. Roy. Statist. Soc., 122A:* 301–324 (1959).

Trocine, L., and L.C. Malone: An Overview of Newer, Advanced Screening Methods for the Initial Phase in an Experimental Design, *Proc. 2001 Winter Simulation Conference,* Washington, D.C., pp. 169–178 (2001).

Tukey, J.W.: *Exploratory Data Analysis,* Addison-Wesley, Reading, Massachusetts (1970).

Turing, A.M.: Computing Machinery and Intelligence, *Mind, 59:* 433–460 (1950).

Van Horn, R.L.: Validation of Simulation Results, *Management Sci., 17:* 247–258 (1971).

Vassilacopoulos, G.: Testing for Initialization Bias in Simulation Output, *Simulation, 52:* 151–153 (1989).

Venkatraman, S., and J.R. Wilson: The Efficiency of Control Variates in Multiresponse Simulation, *Operations Res. Letters, 5:* 37–42 (1986).

Ventana Systems, Inc.: Vensim simulation software, www.vensim.com (2013).

Volterra, V.: Variations and Fluctuations of the Number of Individuals in Animal Species Living Together, in *Animal Ecology*, R.N. Chapman, ed., McGraw-Hill, New York (1931).

von Neumann, J. (Summarized by G.E. Forsythe): Various Techniques Used in Connection with Random Digits, *Natl. Bur. Std. Appl. Math. Ser., 12*: 36–38 (1951).

Wagner, H.M.: *Principles of Operations Research,* Prentice-Hall, Englewood Cliffs, New Jersey (1969).

Wagner, M.A.F., and J.R. Wilson: Graphical Interactive Simulation Input Modeling with Bivariate Bézier Distributions, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 5:* 163–189 (1995).

Wagner, M.A.F., and J.R. Wilson: Recent Developments in Input Modeling with Bézier Distributions, *Proc. 1996 Winter Simulation Conference,* San Diego, pp. 1448–1456 (1996a).

Wagner, M.A.F., and J.R. Wilson: Using Univariate Bézier Distributions to Model Simulation Input Processes, *IIE Trans., 28:* 699–711 (1996b).

Wakefield, J.C., A.E. Gelfand, and A.F.M. Smith: Efficient Generation of Random Variates via the Ratio-of-Uniforms Method, *Statist. Comput., 1:* 129–133 (1991).

Walker, A.J.: An Efficient Method for Generating Discrete Random Variables with General Distributions, *Assoc. Comput. Mach. Trans. Math. Software, 3:* 253–256 (1977).

Wan, H., B.E. Ankenman, and B.L. Nelson: Controlled Sequential Bifurcation: A New Factor-Screening Method for Discrete-Event Simulation, *Operations Res., 54:* 743–755 (2006).

Wan, H., B.E. Ankenman, and B.L. Nelson: Improving the Efficiency and Efficacy of Controlled

Sequential Bifurcation for Simulation Factor Screening, *INFORMS J. Comput., 22:* 482–492 (2010).

Wang, C.-L., and R.W. Wolff: Efficient Simulation of Queues in Heavy Traffic, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 11:* 62–81 (2003).

Wang, C.-L., and R.W. Wolff: New Estimators for Efficient *GI/G/*1 Simulation, *Probability in the Engineering and Informational Sciences, 19:* 219–239 (2005).

Watson, E.F., P.P. Chawda, B. McCarthy, M.J. Drevna, and R.P. Sadowski: A Simulation Metamodel for Response-Time Planning, *Decision Sci., 29:* 217–241 (1998).

Wegman, E.J.: Density Estimation, in *Encyclopedia of Statistical Sciences,* Vol. 2, N.L. Johnson and S. Kotz, eds., John Wiley, New York (1982).

Weiss, W.E.: Development and Use of the Dynamic Security Model for Airports, *J. of Airport Management, 5:* 245–254 (2011).

Welch, B.L.: The Significance of the Difference between Two Means When the Population Variances Are Unequal, *Biometrika, 25:* 350–362 (1938).

Welch, P.D.: On the Problem of the Initial Transient in Steady-State Simulation, IBM Watson Research Center, Yorktown Heights, New York (1981).

Welch, P.D.: The Statistical Analysis of Simulation Results, in *The Computer Performance Modeling Handbook,* S.S. Lavenberg, ed., Academic Press, New York (1983).

Welch, P.D.: On the Relationship between Batch Means, Overlapping Batch Means, and Spectral Estimation, *Proc. 1987 Winter Simulation Conference,* Atlanta, pp. 320–323 (1987).

White, K.P.: An Effective Truncation Heuristic for Bias Reduction in Simulation Output, *Simulation, 69:* 323–334 (1997).

White, K.P., M. J. Cobb, and S.C. Spratt: A Comparison of Five Steady-State Truncation Heuristics for Simulation, *Proc. of the 2000 Winter Simulation Conference,* Orlando, Florida, pp. 755–760 (2000).

Whitt, W.: Bivariate Distributions with Given Marginals, *Annals Statist., 4:* 1280–1289 (1976).

Wichmann, B.A., and I.D. Hill: An Efficient and Portable Pseudo-random Number Generator, *Appl. Statist., 31:* 188–190 (1982).

Wichmann, B.A., and I.D. Hill: Correction to "An Efficient and Portable Pseudo-random Number Generator," *Appl. Statist., 33:* 123 (1984).

Wieland, F.: Parallel Simulation for Aviation Applications, *Proc. 1998 Winter Simulation Conference,* Washington, D.C., pp. 1191–1198 (1998).

Wikipedia: Agent-based model, http://en.wikipedia.org/wiki/Agent-based_model (2013).

Wilensky, U.: NetLogo Wolf Sheep Predation Model, http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, Illinois (1997).

Wilensky, U.: NetLogo, http://ccl.northwestern.edu/netlogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, Illinois (1999).

Wilk, M.B., and R. Gnanadesikan: Probability Plotting Methods for the Analysis of Data, *Biometrika, 55:* 1–17 (1968).

Willink, R.: A Confidence Interval and Test for the Mean of an Asymmetric Distribution, *Commun. Statist.—Theory and Methods, 34:* 753–756 (2005).

Wilson, J.R.: Proof of the Antithetic-Variates Theorem for Unbounded Functions, *Math. Proc. Camb. Phil. Soc., 86:* 477–479 (1979).

Wilson, J.R.: Antithetic Sampling with Multivariate Inputs, *Am. J. Math. Management Sci., 3:* 121–144 (1983).

Wilson, J.R.: Variance Reduction Techniques for Digital Simulation, *Am. J. Math. Management Sci., 4:* 277–312 (1984).

Wilson, J.R.: Modeling Dependencies in Stochastic Simulation Inputs, *Proc. 1997 Winter Simulation Conference,* Atlanta, pp. 47–52 (1997).

Wilson, J.R., and A.A.B. Pritsker: Variance Reduction in Queueing Simulation Using Generalized Concomitant Variables, *J. Statist. Comput. Simul., 19*: 129–153 (1984a).

Wilson, J.R., and A.A.B. Pritsker: Experimental Evaluation of Variance Reduction Techniques for Queueing Simulation Using Generalized Concomitant Variables, *Management Sci., 30:* 1459–1472 (1984b).

Winston, W.L., and S.C. Albright: *Practical Management Science,* 4th ed., Cengage Learning,

Independence, Kentucky (2011).

Wolff, R.W.: Poisson Arrivals See Time Averages, *Operations Res., 30:* 223–231 (1982).

Wolverine Software Corporation: Proof Animation software, www.wolverinesoftware.com (2013).

Wolverine Software Corporation: SLX simulation software, www.wolverinesoftware.com (2013).

Wright, R.D., and T.E. Ramsay, Jr.: On the Effectiveness of Common Random Numbers, *Management Sci., 25:* 649–656 (1979).

Wu, C.F.J., and M.S. Hamada: *Experiments: Planning, Analysis, and Optimization,* 2d ed., John Wiley, New York (2009).

Xu, J., B.L. Nelson, and L.J. Hong: An Adaptive Hyperbox Algorithm for High-Dimensional Discrete Optimization via Simulation Problems, *INFORMS J. Comput., 25:* 133–146 (2013).

Yang, W.-N., and W.-W. Liou: Combining Antithetic Variates and Control Variates in Simulation Experiments, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 6:* 243–290 (1996).

Yang, W.-N., and B.L. Nelson: Multivariate Batch Means and Control Variates, *Management Sci., 38:* 1415–1431 (1992).

Yarnold, J.K.: The Minimum Expectation in χ^2 Goodness-of-Fit Tests and the Accuracy of Approximations for the Null Distribution, *J. Am. Statist. Assoc., 65:* 864–886 (1970).

Yuan, M., and B.L. Nelson: Multiple Comparisons with the Best for Steady-State Simulation, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 3:* 66–79 (1993).

Yuan, M., and B.L. Nelson: Autoregressive-Output-Analysis Methods Revisited, *Ann. of Operations Res., 53:* 391–418 (1994).

Yücesan, E., and L.W. Schruben: Structural and Behavioral Equivalence of Simulation Models, *Assoc. Comput. Mach. Trans. Modeling and Comput. Simul., 2:* 82–103 (1992).

Yücesan, E., and L.W. Schruben: Complexity of Simulation Models: A Graph Theoretic Approach, *INFORMS J. Comput., 10:* 94–106 (1998).

Zanakis, S.H.: Extended Pattern Search with Transformations for the Three-Parameter Weibull MLE Problem, *Management Sci., 25:* 1149–1161 (1979a).

Zanakis, S.H.: A Simulation Study of Some Simple Estimators for the Three-Parameter Weibull Distribution, *J. Statist. Comput. Simul., 9:* 101–116 (1979b).

Zeimer, M.A., and J.D. Tew: Metamodel Applications Using TERSM, *Proc. 1995 Winter Simulation Conference,* Arlington, Virginia, pp. 1421–1428 (1995).

Zeisel, H.: A Remark on AS 183. An Efficient and Portable Pseudo-random Number Generator, *Appl. Statist., 35:* 89 (1986).

Zouaoui, F., and J.R. Wilson: Accounting for Parameter Uncertainty in Simulation Input Modeling, *IIE Trans., 35:* 781–792 (2003).

Zouaoui, F., and J.R. Wilson: Accounting for Input-Model and Input-Parameter Uncertainties in Simulation, *IIE Trans., 36:* 1135–1151 (2004).

中英文名词对照

A

Absolute error(绝对误差)
Acceptance-complement(补选)
Acceptance-rejection method(舍选法)
Accreditation(认证)
Acs1X(一种连续系统仿真软件名称)
Advantages of simulation(仿真的优点)
Agent, definition of(智能体定义)
Agent-based simulation (基于智能体的仿真)
Alias method(别名)
Aliasing of effects(效果的别名)
Analysis of variance(方差分析)
Analytic model(解析模型)
Anderson-Darling test(安德森-达林测试)
Animation(动画)
Antithetic variants(对偶变量)
Anylogic(一种通用的仿真软件包的名称)
Application areas(应用领域)
Application-oriented simulation packages(面向应用的仿真软件包)
AR(autoregressive) process(自回归过程)
Arena(一种仿真软件的名称)
Arena Contact Center Edition, Arena(客服中心版)
ARMA(autoregressive moving average) generation of((自回归滑动平均)的产生)
Arrival process(到达过程)
Arrival rate(到达速率)
ARTA(autoregressive-to-anything) process(任意自回归过程)
ASAP(automated simulation analysis procedure)(自动仿真分析程序)
Assumption document (假设文档)
Attributes(属性)
Autocorrelated(自相关的)
AutoMod(一种仿真软件的名称)
Autoregressive method(自回归方法)
AutoSched(一种仿真软件的名称)
Autostat(AutoMod 支持的软件)
Axial points(轴点)

B

Backlogged excess of demand over supply(需求超过供应的积压量)
Balking(阻塞)
Bank models(银行模型)
Bar charts(条形统计图(棒图)
Batch arrival process(批到达过程)
Batch-means method(批均值法)
Bayesian methods(贝叶斯方法)
Bernoulli distributions(伯努利分布)
Bernoulli trail(伯努利试验)
Beta distribution(β分布)
Beta function(β函数)
Bézier distribution(贝塞尔分布)
Bias(有偏的)
Biased estimator(有偏估计)
Binomial distribution(二项分布)
Bit(二进制位)
Blocked customers(被封锁的顾客)
Blocked experimental designs(带封锁的实验设计)
Boids(群)
Bonferroni inequality, Bonferroni(不等式)
Bootstrapping(步步为营法)
Bottleneck(瓶颈)
Box plot(盒子图)

C

C, C 语言(一种计算机编程语言)
C++, C++ 语言(一种计算机编程语言)
Calibration(校准)
Cauchy distribution(柯西分布)
Center point(中心点)
Central composite design(中心合成设计)
Central limit theorem(中心极限定理)
Chi-square distribution(χ^2 分布)
Chi-square test(χ^2 检验)
Code reusability(代码可重用性)
Code variables(代码变量)
Coefficient of variation(方差系数)
Combat models (对抗模型)
Combined discrete-continuous simulation(组合式离散-连续仿真)
Combined multiple recursive random-number generators(组合式多重回归随机数发生器)
Common random numbers(公共随机数)

Communications networks(通信网络)
Comparing alternative system configurations(比较多种系统配置)
Complex adaptive system(复杂自适应系统)
Composite random-number generators(组合随机数发生器)
Composition(组合)
Compound Poisson process(复合泊松过程)
Computer models(计算机模型)
Conceptual model(概念模型)
Conceptual model validation(概念模型确认)
Conditional distribution(条件分布)
Conditional Monte Carlo Method(条件蒙特卡罗法)
Conditional variance(条件方差)
Conditioning for variance reduction(调节方差减少)
Confidence intervals(置信区间)
Confidence intervals-(Cont.)(置信区间-(恒量))
Confounding of effects(效果的混淆)
Conservation equations(守恒方程)
Conservative synchronization(保守同步)
Continuous distribution(连续分布)
Continuous simulation(连续仿真)
Continuous system(连续系统)
Continuous-time statistics(连续时间统计)
Contour plot(等值线图)
Control variates(控制变量)
Conveyors(传送带)
Convolution(卷积)
Correlated sampling(相关采样)
Correlated variates(相关变量)
Correlation(相关性)
Correlation function(相关函数)
Correlation plot(相关图)
Correlation test for random-number generator(随机数发生器的相关性测试)
Cost module(费用模块)
Covariance(协方差)
Covariance matrix(协方差矩阵)
Covariance-stationary process(协方差稳态过程)
Coverage (覆盖度)
Credibility(可信性)
Crystal Ball(水晶球(预报未来的方法))
Cube plot(立方图)
Cumulative distribution function(累计分布函数)

D

Data collection(数据收集)
Debugging (See Verification)(调试)
Decision Variables(决策变量)
Degree of freedom(自由度)
Delay in queue(在队列中的延迟)
Deletion of output data(输出数据的删除)
Delivery lag(交货延期)
Density function(密度函数)
Density-histogram plots(密度直方图)
Dependence(隶属)
Dependent(隶属的)
Design-Expert(设计专家)
Design matrix(设计矩阵)
Design points(设计点)
Detail(细节)
Deterministic simulation model(确定性仿真模型)
Differential equations(微分方程)
Digamma function(双伽马函数)
Disadvantages of simulation(仿真的缺点)
Discrete distributions(离散分布)
Discrete-event simulation(离散事件仿真)
Discrete simulation model(离散仿真模型)
Discrete system(离散系统)
Discrete-time statistics(离散时间统计)
Discrete uniform distributions(离散均匀分布)
generationfrom(由(离散均匀分布)产生)
Distributed simulation(分布仿真)
Distribution function(分布函数)
Distribution-function-differences plot(分布函数差图)
Documentation(文档)
Dot plot(点图)
Double-exponential distribution(双指数分布)
Double-precision arithmetic(双精度算法)
Doubly linked list(双链表)
Dymola(一种仿真软件的名称)
Dynamic simulation model(动态仿真模型)
Dynamic system behavior(动态系统行为)

E

EAR processes(EAR 过程)
Ease of use(易用)
Economic systems(经济系统)
Emergence(应急)
Empirical distribution(经验分布)
Encapsulation(封装)
Enterprise Dynamics(企业动力学)
Entity(实体)
Erlang distribution(厄兰分布)
Estimating parameters(参数估计)
Estimator(估计方法)
Event(事件)
Event graphs(事件图)
Event list(事件表)
Event routine(事件例程)

F

G

H

I

J

K

L

M

Monte Carlo simulation(蒙特卡罗仿真)
Moving average(滑动平均)
MSER(最大稳定极值区域(Maximally Stable Extremal Regions)
Multiple measures of performance(性能的多重度量)
Multiple recursive random-number generators(多重回归随机数发生器)
Multiple-comparisons problem(多重比较问题)
Multivariate distributions and random vectors(多变量分布与随机向量)

N

Natural variables(固有变量)
Negative binomial distribution(负二项分布)
Negatively correlated random variables(负相关随机变量)
NetLogo(一种复杂系统仿真软件)
Neural networks(神经网络)
Newton's method(牛顿法)
Next-event time advance(下一时间推进)
Nonhomogeneous Poisson process (非均匀泊松过程)
Nonstationary(非平稳)
Nonstationary Poisson process(非平稳泊松过程)
Nonterminating simulation(非终止型仿真)
steady-state cycle parameter(稳态周期参数)
steady-state parameter(稳态参数)
Normal distribution(正态分布)
NORTA (normal-to-anything) random vectors((正态到任意)随机向量)
Nugget parameter(块参数)
Number of replications(重复运行的次数)

O

Object-oriented simulation(面向对象的仿真)
One-factor-at-a-time approach (OFAT)(一次一因子法)
OPENT Modeler(OPENT 建模器(一种用于通信网络的面向应用的仿真软件包))
Optimistic synchronization(优化同步)
Optimization(优化)
OptQuest(仿真软件 ARENA 的一个优化模块的名称)
Order statistics(序统计)
Output data analysis(输出数据分析)
Output reports(输出报告)
Overflow, integer(溢出,整数)
Overlapping batch means (重叠批均值,参见 Batch-means method)

P

Paired-t confidence interval(双-t 置信区间)
Parallel simulation(并行仿真)
Parameterization of distributions(分布的参数化)
Pareto distribution(帕雷托分布)
Peak-load analysis(峰荷分析)
Pearson type Ⅴ distribution(皮尔逊类型Ⅴ分布)
Pearson type Ⅵ distribution(皮尔逊类型Ⅵ分布)
Percentile (百分数)
Performance measures (性能度量)
Perishable inventory(有保存期的库存)
Phase plot(条形图)
Phi-mixing(ϕ 混合)
Pie charts(饼图)
Pilot runs(控制运行)
Pitfalls in simulation(仿真的缺点)
Plackett-Burman designs(筛选试验设计)
Point estimator(点估计,参见 Estimator, point)
Poisson distribution(泊松分布)
Poisson process(泊松过程)
Polymorphism(多态性)
Positively correlated random variables(正相关随机变量)
Powersim(一种仿真软件)
P-P plots(P-P 图)
Practically significant difference between model and system(模型与系统间的显著差别)
Predator-prey models(捕食模型)
Predecessor link(前链)
Primitive element(素元)
Primitive over Galois field(伽逻瓦域上的素元)
Priority queue discipline(优先排队规则)
Probability density function(概率密度函数)
Probability mass function(概率质量函数)
Probability plots(概率图)
Problem of the initial transient(初始瞬态问题)
Process approach(进程法)
Process, defined(进程,所定义的(进程))
ProcessModel(过程重构或服务领域的一种面向应用的仿真软件包的名称)
Production runs(生产运行)
Production scheduling(生产调度)
Programming time(编程时间)
ProModel(制造领域的一种面向应用的仿真软件包的名称)
Proof 3-D Animation(三维动画领域的一种面向应用的仿真软件包的名称)
Pseudorandom numbers(伪随机数)

Q

Q-Q plots(分位数-分位数图)
Quadratic congruential random-number generators

R

S

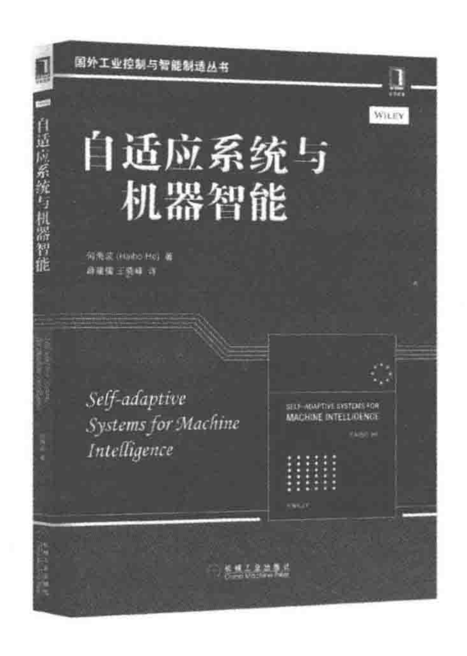
T

U

V

W

推荐阅读



自适应系统与机器智能

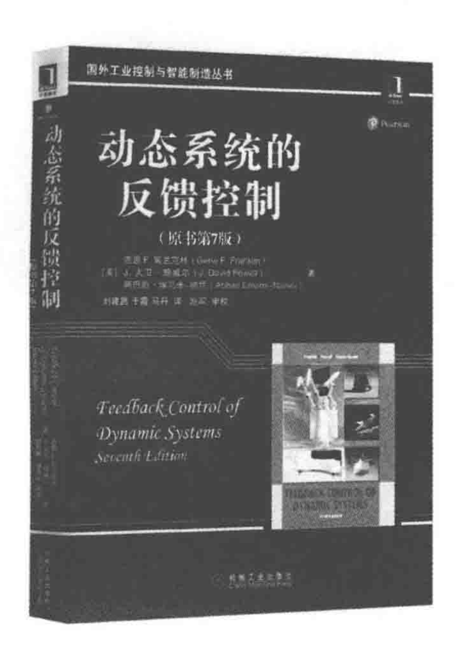
书号：978-7-111-54114-1 作者：何海波（Haibo He） 译者：薛建儒 等 定价：59.00元

机器智能研究是关于自适应系统的原理、基础和设计的研究，这种自适应系统能够学习、预测和优化，并通过与不确定的环境交互做出决策，从而完成系统目标。本书有助于对自适应智能系统的基本理解，促进读者向模拟某些类脑智能水平的长期目标前进，同时也使如今许多复杂系统的智能水平更接近现实。

本书分为以下4个主要部分

- 第一部分介绍了用于机器智能研究的自适应系统，给出了研究的意义以及传统计算机与类脑智能的主要区别。
- 第二部分重点讨论了机器智能研究的数据驱动方法，着重介绍了增量学习、不平衡学习和集成学习。
- 第三部分着重介绍机器智能研究的生物启发式方法，详细讨论了自适应动态规划、联想学习和序列学习。
- 第四部分简要介绍机器智能关键硬件的设计，如功耗、设计密度、内存和速度，目的是实现大规模、复杂的综合智能硬件系统。

推荐阅读



动态系统的反馈控制（原书第7版）

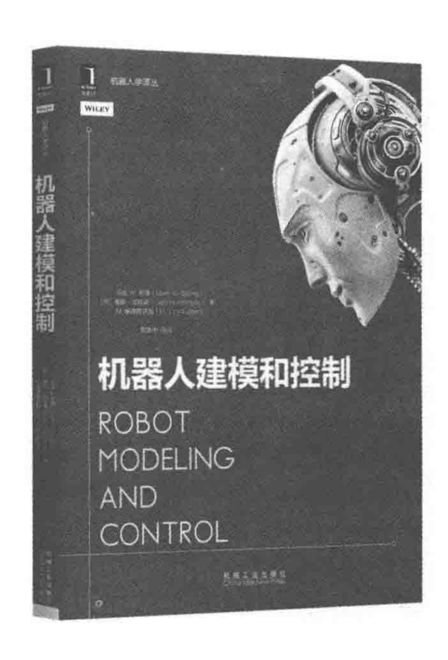
书号：978-7-111-53875-2 作者：吉恩 F. 富兰克林 等 译者：刘建昌 等 定价：119.00元

本书系统阐述了反馈控制的基本理论、设计方法及在工程技术领域中的一些实际问题。本书主要利用根轨迹、频率响应和状态变量方程这三种方法将控制系统的分析与设计结合起来，并结合大量实例和Matlab代码进行控制系统的分析与设计，来阐述相关内容。第7版更加注重反馈控制在整个控制理论体系中的地位，因而对部分章节进行了调整，对书中的实际例子进行了更新，而且对各章的习题进行了修正和补充。

本书特色

- 本书融合了作者多年的教学经验，结合工程实际应用，循序渐进地对基本理论和设计方法进行介绍，尤其对控制系统的分析与设计具有非常高的参考价值。
- 每章开篇增加了展望和综述，从科学的角度展开论述该章的内容，便于读者对知识的理解和掌握。
- 每章末尾的小结归纳了该章的关键概念和结论，有助于读者抓住重点。
- 每章末尾都有复习题，便于学生自学。
- 生成算法。
- 采用基础和高等两种方法对机器人运动和力控制进行综合处理。
- 对几何非线性控制的处理比其他高等教材更加易懂。
- 通过大量的实例和完整的习题阐明了理论的各个方面。

推荐阅读



机器人建模和控制

书号：978-7-111-54275-9 作者：马克 W. 斯庞 等 译者：贾振中 等 定价：79.00元

本书所覆盖的内容在深度和广度方面都是独一无二的。据我所知，没有其他教材能够对现代机器人的操作和控制做出如此精彩而又全面的概述。

—— 布拉德利·毕晓普（Bradley Bishop），美国海军学院

本书由Mark W. Spong、Seth Hutchinson和M. Vidyasagar三位机器人领域顶级专家联合编写，全面且深入地讲解了机器人的控制和力学原理。全书结构合理、推理严谨、语言精练，习题丰富，已被国外很多名校（包括伊利诺伊大学、约翰霍普金斯大学、密歇根大学、卡内基-梅隆大学、华盛顿大学、西北大学等）选作机器人方向的教材。

本书特色

- 通过循序渐进的计算方法帮助你推导和计算最通用机器人设计中的运动学正解、运动学逆解和雅克比矩阵问题。
- 详细覆盖了计算机视觉和视觉伺服控制，使你能够通过对带有相机感知元件的机器人编程来操作物体。
- 通过一个完整章节的动力学讲解，为计算最通用机械臂设计中的动力学问题做好准备。
- 初步介绍了最通用的运动规划和轨迹。

本书被誉为仿真界的 “圣经”。自从1982年第1版出版以来，本书一直致力于揭示综合的仿真知识、最新的仿真进展和专业化的仿真方法。本书借助大量直观的例子、图和习题增加了可读性，很适合作为大学教材、仿真操作指南和自学读物。

本书特色

- 新增了基于agent仿真和系统动力学的一章，这些内容在仿真领域广泛使用。
- 进一步详细讨论了经典的实验设计，特别强调了最适合仿真的设计和元模型。
- 介绍了用于确定仿真预热周期和估计仿真系统稳态均值的最新统计方法。
- 讲述了在不同系统配置下比较性能的最新方法。
- 展示了目前非常新颖的仿真软件。
- 回顾了基本概率和数理统计知识。
- 涵盖了大量例子、图和习题。

作者简介

埃弗里尔 M. 劳（Averill M. Law）在加州大学伯克利分校获得工业工程与运筹学硕士和博士学位，在加州大学长滩分校获得数学硕士学位，在宾夕法尼亚州立大学获得数学学士学位。他是Averill M. Law & Associates公司（亚利桑那州图森市）总裁，是许多机构的仿真顾问，是ExpertFit分布拟合软件的开发者。他在19个国家讲过500多个仿真与统计方面的短期课程。



www.mheducation.com

投稿热线：(010) 88379604
客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

封面设计·锡 彬



上架指导：仿真建模

ISBN 978-7-111-55746-3



定价：119.00元

[General Information]
书名=仿真建模与分析　原书第5版
作者=（美）埃佛里尔.M.劳（Averill
页数=498
SS号=14163081
DX号=
出版日期=2017.01
出版社=机械工业出版社